Agent-Based Lost Person Movement Modelling, Prediction and Search in Wilderness

Wallizada Mohibullah

A dissertation submitted for the degree of

Doctor of Philosophy

of the

University College London.

Department of Computer Science

University College London

February 20, 2017

I, Wallizada Mohibuillah, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

In this research we investigate the problem of searching for a Lost Person (LP) in wilderness using an autonomous Unmanned Aerial Vehicle (UAV). The problem of search with a UAV is often treated as gridded environment search where the state of each grid (cell) is examined individually for the presence or absence of the target. However, this idealised way of search fails to exploit many potentially valuable dependencies and secondary cues — such as material deposited or left by the LP or topographical features such as natural tracks (trails) — which could significantly aid the search process.

We discuss the need for such a system and review the current state-of-the-art work. Since key to a quick and successful search is a well defined initial distributions. We further argue the need to generate the initial distribution over the trajectory of the LP, not merely the end location, usually done in literature.

We propose a search framework consisting of three key phases: *information gathering*, *initial distribution generation* and *search*. In the information gathering phase, we collect detailed information related to both the LP and the search environment. Then in the initial distribution generation phase, using the information gathered, we generate distribution over the LP's trajectory using particles. Each particle represented by an agent model of LP movement with sampled parameters, navigating and interacting with the environment represented using data-sets in the form of terrain elevation, topography and vegetation. To ensure, the agent model is a good representation of the LP behaviour, we calibrate its parameters using the method called SMC^2. Finally in the *Search* phase, a UAV is deployed to explore the search area and detect the LP, any evidence features or changes in the environment. All information detected are localised and used to update the distribution over the LP trail until either the LP is located or the search is terminated.

Acknowledgements

I would like to dedicate my PhD to my late mother whom I love dearly. In addition, I would like to convey my deepest gratitude to my uncle (Nusrullah Wallizada) who has always encouraged me through hardships and has been the source of my inspiration. A special thanks to my wife who has shown extreme patience and has always been very caring and my lovely daughter (little Issra Mohib) who has been the joy of our life.

I would also like to take this opportunity to thank my supervisors Dr. Simon Julier and Professor Steven Hails who have been an incredible guide and support and source of encouragement throughout my time as a research student. In particular I would like to thank Dr. Simon Julier, who has been instrumental in my progress and has always been accommodating, offering advice, support and feedbacks on every aspect of my research, even during holidays.

I would also like to thank Dr. Renzo Denardi, Dr. Luke Teacy and Dr. Timothy Patterson for their support and guidance during experiments.

Other Acknowledgments My research was supported under the EPSRC-funded project "SUAAVE: Sensing Unmanned Autonomous Aerial VEhicles" (EP/F064179/1).

This project was concerned with the creation, control and evaluation of swarms of autonomous UAV for a wilderness search and rescue type scenarios.

Contents

List of Figures

List of Tables

Acronyms

ABM Agent-Based Modelling

AHP Anchored Homogeneous Point

DEM Digital Elevation Model

DI Delayed Initialisation

DOE Design Of Experiment

DSM Digital Surface Model

DTM Digital Terrain Model

EMD Earth Movers Distance

EKF Extended Kalman Filter

FOV Field Of View

GPS Global Positioning System

IDP Inverse Depth Parameterisation

IMU Inertial Measurement Unit

INS Inertial Navigation System

KF Kalman Filter

KLD Kullback Leibler Divergence

LP Lost Person

MCMC Markov Chain Monte Carlo

NEES Normalised Estimation Error Squared

OSPA Optimal Sub-Pattern Assignment

PDF Probability Distribution Function

PF Particle Filter

PLS Point Last Seen

SFM Social Force Model

SLAM Simultaneous Localisation and Mapping

SMC Sequential Monte Carlo

SUAAVE Sensing Unmanned Autonomous Aerial VEhicle

UAV Unmanned Aerial Vehicle

UTM Universal Transverse Mercator

VCL Visual Critical Length

WiSAR Wilderness Search and Rescue

Chapter 1

Introduction

1.1 Introduction

This chapter introduces the research problem addressed in this thesis – modelling the search framework for automation of search process in wilderness areas. We begin by presenting the context for this research in Section 1.2, where we detail the wilderness search and rescue steps, how remotely controlled Unmanned Aerial Vehicles (UAVs) are used to assist in the search process and discussing the need for autonomous Unmanned Aerial Vehicles (UAVs) with improved search capability. Then in Section 1.3 we list the main contributions of this research in incorporating autonomous Unmanned Aerial Vehicles (UAVs) in the wilderness search process. Finally in Section 1.3, we detail the structure of the thesis with the help of a flow diagram.

Key terms used in this chapter

- **LP**: This is acronym for the lost person.
- **WiSAR**: This refers to the manual search and rescue in wilderness.
- **UAV**: This is acronym for unmanned aerial vehicle.

1.2 Context

1.2.1 Wilderness Search and Rescue

Outdoor wilderness activities, such as hiking are a major source of recreational activities. Between 2005 and 2008, around 70 million people aged 16 or older visited wilderness or other wild-land areas and a further 200 to 300 million visited national parks [1].

Unfortunately, people can go missing, and need to be rescued. In the US National Parks alone, between 1992 and 2007, there were 78,488 individuals involved in 65,435 search incidents, costing almost \$60M [2]. Of these, 3.39% ended up in fatalities and 30.94% in personal injuries to the victims. There are several ways in which people go missing. These include people being unable to move (fatigue, health, lack of light) or getting lost.

Table 1.1: Statistics on the lost hiker survivability in wilderness [3]. $\mathbf{n}$ is the number of LP cases for each survivability category.

Survivability		
Category	**Found in Wilderness**	
Uninjured	78%	
Injured	16%	
Fatality	6%	
Survivability	**Alive**	**n**
$<$ 24 hours	97%	2460
24-48 hours	76%	361
48-72 hours	60%	118
72-96 hours	52%	51
$>$ 96 hours	49%	23

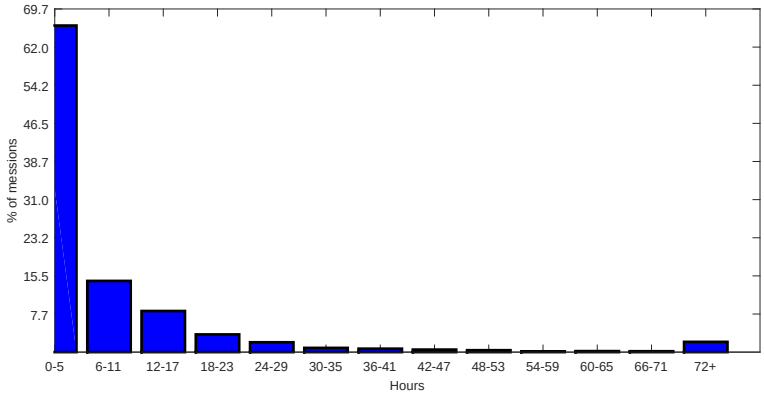


Figure 1.1: Time required to complete a WiSAR mission based on 12,900 cases.

Given the risks and dangers involved, many park services have process and procedures for addressing this problem. The process used by land search teams in wilderness areas is called Wilderness Search and Rescue (WiSAR). Key to a successful WiSAR mission is the search team's ability to locate the LP as quick as possible. Table 1.1 illustrates the importance of search times. It tabulates survivability in terms of the search time before lost hikers are found. It shows that there is a negative correlation between finding a Lost Person alive and the search times. The fact that 16% of the hikers suffered from injuries, with an additional 6% fatalities, makes the case for quick search and rescue to ensure safe rescue of the LP. Figure 1.1 illustrates the search times for 12,900 search cases [3]. As we can see, wilderness search can take anything between hours to days.

Having determined the size of the search area where the LP is believed to be in, the three key phases of WiSAR are:

- *Information gathering*: In this phase, information about both the LP and the search area are compiled.

- *Initial distribution generation*: In this phase, the search area is first segmented into regions based on information such as topography of the area, elevation changes and LP preferences, and then using the information gathered, a belief value map of finding the LP in each of the segments is specified.

- *Search*: Using the initial distribution, the search is initiated by assigning teams of searchers to segments. During the search operation, observations are made of the environment, that are used to update the belief value map.

Traditionally, these phases have been carried out manually using teams of people. This can be slow, dangerous, expensive and difficult due to the typical large size of wilderness search areas, their ruggedness and remoteness [4], and the unfamiliarity of the search teams with the area. Methods to decrease search times and improve rescuers' safety are extremely important. One potential approach is to use manned aerial vehicles to complement ground searches. This is due to the fact that they:

- *Are faster than most human responders:* They can cover large areas very quickly.

- *Can avoid environmental hazards:* They makes it easier to traverse over challenging terrain types (e.g. aerial vehicle fly over a swiftly flowing river, whereas human rescuers would have to ford it).

- *Can act as an eye-in-the-sky:* They can relay real-time aerial imagery and other sensory data, allowing for fast response to situations such as detection of the LP.

Although they have many advantages, manned aerial vehicles can be costly to operate. For example, in Elias National Park and Preserve in the US, ground search costs range from $18,000 to $29,000 per operation [2]. However, if aerial vehicles are used, cost can be significantly higher. Helicopter costs $1,600 per *hour*. More specialised assets, such as the C-130 turbo prop can be more than $7,600 per hour.

Moreover, flying over rugged terrain often involves dramatic terrain changes, up-drafts and downdrafts, which make search by both manned fixed-wing aerial vehicle and helicopters dangerous [5].

As a result, a natural question to ask is whether Unmanned Aerial Vehicles (UAVs) can be used instead.

1.2.2 Unmanned Aerial Vehicles

A UAV refers to a class of aerial vehicles with no pilot on board. UAVs can be either *remotely piloted* or *self piloted* [6]. Remotely piloted UAVs are controlled by a human operator on the

Figure 1.2: A remotely piloted UAV used in a search and rescue [7].

ground. Self piloted UAVs, also called *autonomous* UAVs, fly independent of any operator feedback or intervention. They choose how to operate on moment-by-moment basis in order to achieve a set of high level goals e.g. decisions like where to go next to look for a target. They operate by following a set of given navigational way-points or a more complex dynamic control systems. To navigate and investigate, both types of UAVs are equipped with navigational sensors such as Inertial Measurement Unit (IMU) and Global Positioning Systems (GPSs), processing boards for functions like image processing, and observation sensors such as a visual camera. Figure 1.2 shows a remotely piloted UAV used in a search and rescue operation. Although the flight time of a UAV on any single flight may be considerably lower than that of a manned aerial vehicles (typically no more than 20 minutes), UAVs have many advantages over manned aerial vehicles. For example they are:

- *Small*: This property enables UAVs to investigate hard-to-reach areas.
- *Expendable*: If one is lost or damaged, it is not a tragedy and can simply be fixed or replaced.
- *Inexpensive*: A typical UAV costs around $1000 to purchase, compared with the much more expensive per hour cost of a helicopter.
- *Safer*: It decreases risk to personnel (pilots and search teams).

In WiSAR, remotely piloted UAVs are mainly used to assist in search. A good example that illustrate all the advantages of using UAVs in search and rescue missions occurred in the

US in July 2014. In this search and rescue mission, a remotely piloted UAV was used to locate a person that had been lost for three days [8,9]. After a three day search with a helicopter, search dogs and hundreds of volunteers, a UAV was deployed to look at a 200-acre bean field, from about 200 feet in the air. As a result, in just 20 minutes the UAV had managed to comb through most of the field and locate a man later identified as the LP, standing in the middle of the field. An example like this shows that UAVs can have a big impact on WiSAR.

1.2.3 Use of Remotely Piloted Unmanned Aerial Vehicles in WiSAR

Considering the advantages of UAVs, there has been a great deal of research into incorporating UAVs into search and rescue process [5, 10–20].

The seminal work by Goodrich et al. in [10, 12, 14, 21] investigated the use of UAVs in various WiSAR scenarios. They performed a number of field tests, analysed the problem identified the critical steps, and proposed a set of different search paradigms (Appendix B.2) in which UAVs can be used to search for a lost target.

In UAV assisted WiSAR, similar to manual WiSAR, the search is directed by an incident commander who coordinates activities of various search teams. The only addition to the team is the technical team, which includes:

- A UAV operator who remotely pilots the UAV and guides it to a series of locations that allow the camera to obtain imagery of potential evidence features related to LP movement.
- One or more sensor operators, who control and direct the sensors. For example, directing a gimballed camera to capture part of the search area that is likely to contain evidence. Additionally, they are responsible for analysing, inspection and interpretation of imagery to detect potential evidence related to the LP.

The search operation is initiated by deploying a UAV to help three specific objectives:

- To quickly cover the search area.
- To gather evidence from a safe distance.
- To detect or locate the LP.

These field tests however, consider UAVs in a *remote piloted* mode. They lack autonomy. As mentioned earlier, teams of people are required for various roles, such as control of the UAV, control of the sensor and spotting of the evidence features [15].

Furthermore, human operators normally have limitation in both data analysis and in controlling the UAV due to reasons such as:

- *Mis-communication between operators:* Information loss or misunderstandings in communication between operators controlling different aspects of search due to the pressure of the situation, which could result in concerns like risk of collision, flying over the same area multiple times, etc.

- *Information overload:* Trying to control and monitor too many things at once.

- *Operator fatigue:* Becoming tired after some time of operating or looking for evidence features.

- *Coordination difficulty:* Trying to establish the coordination of mental models and activities.

These limitations restrict the operational feasibility and efficiency of remote piloted UAVs, especially if a swarm (collection) of UAVs are used to cover large areas of the wilderness in short time, which in the case of search and rescue operations really improve the chances of locating a Lost Person faster, thus improving chances of a rescue.

Autonomous systems and automated search processes have the potential to address these limitations. Therefore, the Sensing Unmanned Autonomous Aerial VEhicle (SUAAVE) project, within which the work in this thesis was undertaken, was directed towards this outcome.

1.2.4 The Sensing Unmanned Autonomous Aerial Vehicle Project

The goal of the Sensing Unmanned Autonomous Aerial VEhicle (SUAAVE) project, funded as part of the ESRC WINES-II and ran between 2008 and 2012, was to explore how mini UAVs such as the one in Figure 1.2 could be used to aid search and rescue [22]. It involved teams from University College London, University of Ulster and University of Oxford. Figure 1.3 shows the search process flow diagram used in the project. This included:

- Object detection: This is perhaps the most important task in UAV assisted search, where the UAV is given the capability to detect an object of interest (the LP or evidence features in our case) from an altitude. This capability required developing observation models for the platform that consider various factors such as the search platform's altitude, land cover and weather conditions in detection of the object. To address this, research was performed by the teams from Ulster and Oxford [18, 23].

- UAV search path planning: This relates to computing UAV path in an unknown and uncertain environment. This research was performed by the UCL team [24].

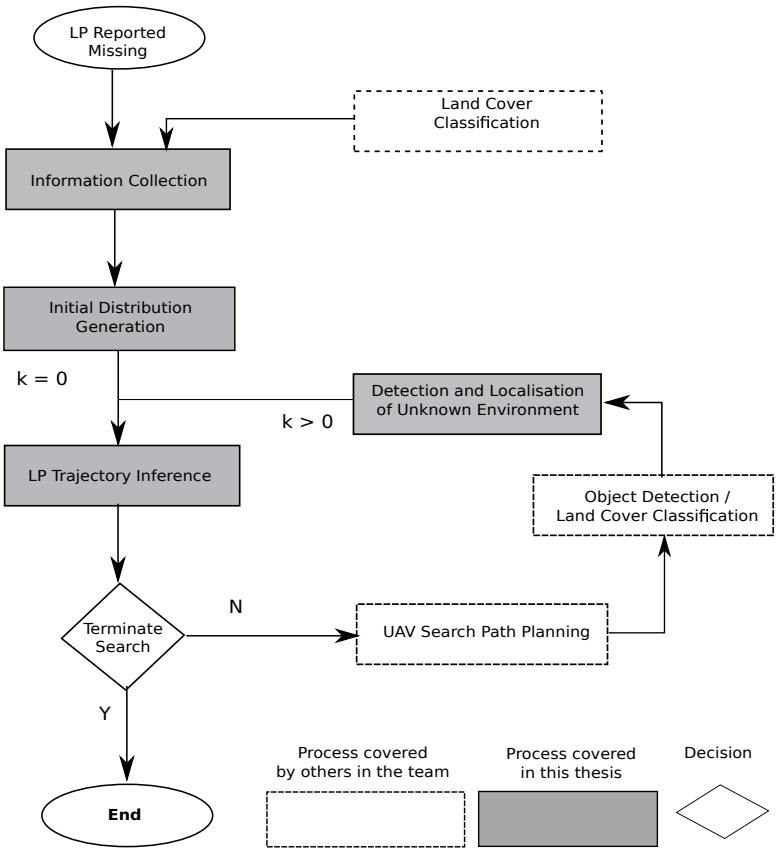


Figure 1.3: Flow diagram of automated search process.

- Land cover classification: This relates to classifying images of search area in terms of topography and vegetation for both safe landing area detection and initial distribution generation. This research was performed by the team in Ulster [25–28].

- UAV Simulator: Since it was not possible to perform experiments using real platforms, both due to cost and safety, a realistic UAV simulator was required. This was developed by the UCL team [29].

With all the UAV-related tasks covered, this research is aimed at modelling the overall search framework: the initial distribution generation based on knowledge of LP behaviour and environment, and inference of the LP trail based on new localised observations of the environment.

The platform used in SUAAVE project is the Ascending Technologies Pelican quad-rotor helicopter shown in Figure 1.4 [30].

The Ascending Technologies platform was selected because it is agile, easy to fly and safe to operate. This platform has flight speed of around $10m/s$ with approximately 20 minutes battery life. The Pelican is fitted with a colour camera, an IEEE 802.11 wireless networking card, a GPS receiver, an Inertial Navigation System (INS) and an ATOM PC board enabling automated in-flight processing and analysis of captured colour aerial imagery. The path of the

Figure 1.4: The UAV used in our project. It is a modified Pelican developed by Ascending Technologies [30] . It includes an on board computing platform together with a camera.

UAV is specified as a series of way points. Without vision-based aiding, its position can be measured with an accuracy of 1.5–2.5m.

1.3 Overview of the Thesis

This thesis is concerned with the creation of new search framework that considers probabilistic techniques to help plan the UAV's movement and to maintain a representation of the spatial uncertainty of the LP. Overall aiming to reduce search times with respect to current search methods.

Since we are dealing with a relatively information sparse environment, modelling plays a very large role in the performance of an estimation algorithm. Models of LP movement show that global characteristics and influences must be taken into account. Therefore, we develop methods which allows us to estimate the entire trail of the LP, and not merely the end location, which is the norm in search literature.

Following the search process illustrated in Figure 1.3, we investigate and develop algorithms for modelling the *initial distribution generation* and *search* phases of the search operation.

To generate the trail-based initial distribution, we investigate and develop a novel agent model of LP movement considering things like the LP's profile, state dependent behaviour and interaction with environment, both local and global. The generated initial distribution is then used to prioritise a search area for the initiation of the search phase.

To ensure the developed agent model reflects a LP behaviour, we tune its parameters using

observed data. Since the number of parameters can be quite high, we use a two step process: sensitivity analysis of parameters to identify significant parameters and then a calibration process using a novel method to tune the identified parameter values.

Next, in the search phase, upon deploying a UAV to search the environment, and receiving new observations, we use them to recursively update the distribution over the LP trail. Compared with current search literature, the crucial difference in the update model is that we consider three different information types: observation of the LP, evidence related to the LP and new observations of the land cover.

However, before using these information types in the update of distribution over trajectory, they need to be localised. Therefore, we investigate different localising methods and use one that produces consistent accurate results to localise the UAV and UAV observations.

1.3.1 Scope of the Project

Automated WiSAR is a large and complicated process. In this thesis we consider a restricted subset of the tasks within the process. Therefore, we make the following assumptions to limit the scope:

- *Nature of the Search area:* People often go missing in different types of wilderness terrain e.g. mountainous, desert, thick forests. For this research we consider areas where there is not much which occludes seeing the LP from the air e.g. flat open grassland.

- *Number of UAVs:* Many applications have shown that multiple UAVs improve the performance of an overall search. However, this requires a suitable design of information sharing mechanisms [31] which we did not have time to develop in this thesis. Therefore, we only consider the case in which a single UAV is in operation at any given time.

- *The LP Type:* More than 33 different types of LP categories have been identified [3]. Each type of LP exhibits different responses to the environmental and topographical situations. For example, a despondent person does not wish to be found. Therefore, they will actively hide from search activities. In this research we only consider Hikers – the dominant category of lost people. Hikers do not deliberately try to hide. Rather, they try to stay clear of thick vegetation, stay on clear ground where possible and follow linear features. More details about the Hiker class can be found in [3].

- *Number of LPs:* We assume only one hiker is missing.

- *Closed world assumption:* It is assumed that the LP is guaranteed to be somewhere inside the search area.

(a) Water Bottle

(b) Piece of cloth

(c) A Glove

(d) A paper bag

Figure 1.5: These images show some of the objects lost people can leave behind and can potentially be detected.

- *Types of evidence:* As the LP moves through the environment, they can leave two types of evidence: changes to the search area as a result of their movement and items deposited by them. In this thesis, we only consider evidence deposited by the LP. Examples of these are shown in Figure 1.5. These are relatively easy to detect (depending on the terrain and search area features). We also assume that due to the nature of the wilderness environment, not many people would traverse it, therefore there are not many false positives (litter left by other people).

- *Object Detection and Classification:* Because we are dealing with the general search model in this research and also because the land cover classification is done by our colleague[1], we will not be covering object detection and land cover classification in this thesis. Throughout the project we cooperated with our colleague for classification and detection of land cover and paths and consider evidence features deposited as point features.

- *Light and Weather Conditions:* The duration of a typical search mission can be many hours. However, in this research we assume sufficiently short search times in which the light and weather conditions can be assumed to remain unchanged.

- *Obstacles and Places of Interest:* Because we will use data sets to represent the search area, we will only consider obstacles and places of interest that can be classified from the data sets or from the UAV observations of the search area during the search operation.

1.3.2 Main Contributions

The key contributions of this thesis within the context of UAV based wilderness search for a Lost Person are:

[1] It is covered by our affiliate Timothy Paterson in the SUAAVE project in [28]

1. A novel agent model for human movement constructed using a profile of the LP and their interaction with the environment. This model incorporates local and global effects of the environment, perception and orientation strategies on the LP's movement. We argue and demonstrate that in contrast to current human movement models, our proposed agent model better reflects a Lost Person's movements, which can have a significant positive effect on the search process. This research formed the basis for the following publication:

   Mohibullah, W., and Simon J. Julier. "Developing an Agent Model of Missing Person in Wilderness." In System Man and Cybernetics (SMC), 2013 Conference on, pp. 1-8. IEEE, 2013.

2. A Bayesian approach to exploit secondary information about the passage of a LP, such as evidence features deposited, in the update of the initial distribution over the LP trail. We argue that a path-based LP trail representation can improve our estimate of the LP's whereabouts. We demonstrate the performance of the algorithm in a simple search scenario, and show a significant improvement over current search methods. This research formed the basis for the following publication:

   Mohibullah, W., and Simon J. Julier. "Stigmergic search for a lost target in wilderness." In Sensor Signal Processing for Defence (SSPD 2011), pp. 1-5. IET, 2011.

3. A study on the effects of a range of conditions including target location, nadir angle of the camera and the trajectory of the UAV on localisation and map building. This research formed the basis for the following publication:

   Mohibullah, W., and Simon J. Julier. "Bearings-only localisation of targets from low-speed UAVs." In Information Fusion (FUSION), 2010 13th Conference on, pp. 1-8. IEEE, 2010.

1.4 Structure of the Thesis

This thesis is divided into eight chapters. The diagram in Figure 1.6 is an expansion of Figure 1.3 and shows how these chapters map into the WiSAR search process developed for SUAAVE.

In Chapter 2, we present a review of current literature on search models considered state-of-the-art. Based on critical analysis, we propose an improved search model in Chapter 3. This model is composed of two key phases: *initial distribution generation* and *search* phases. To compute the initial distribution, we probabilistically model the distribution over LP trail. We

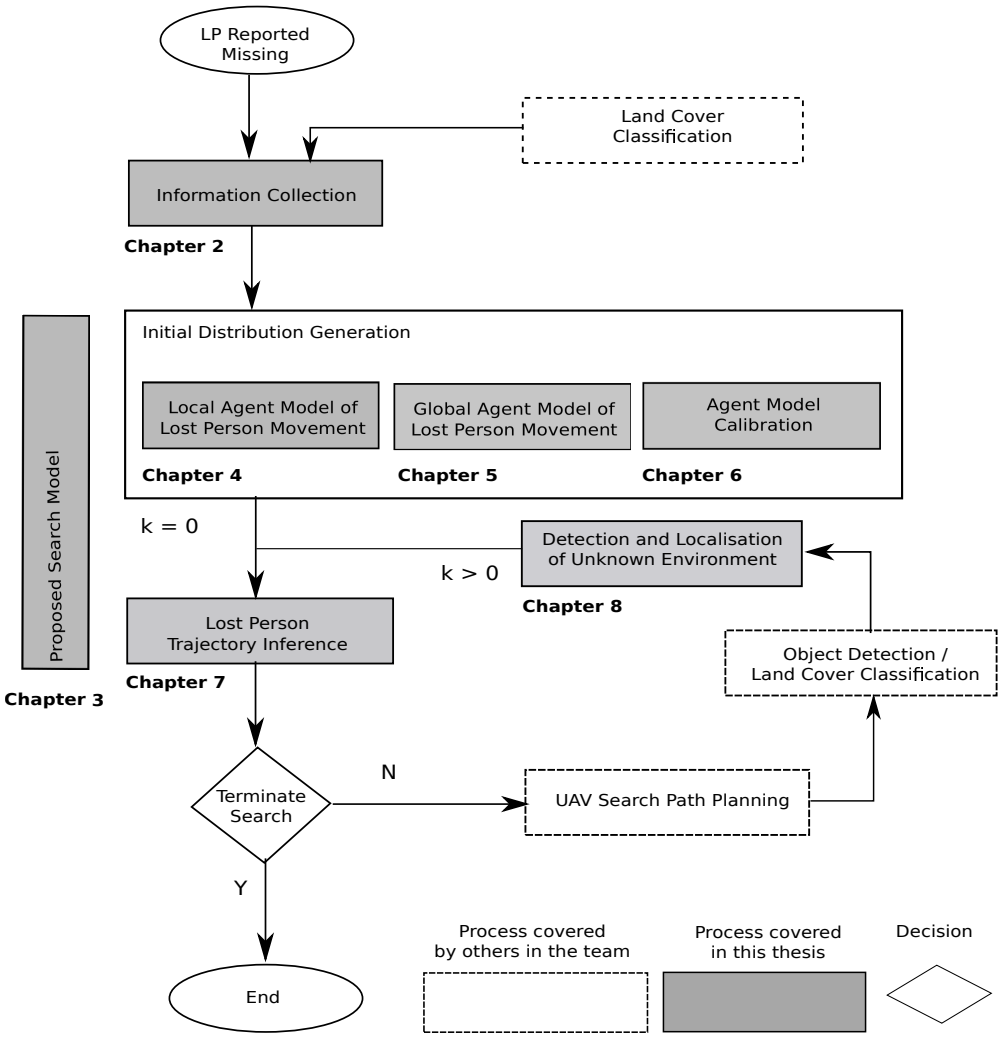


Figure 1.6: The flow diagram showing the search phases/processes and their coverage in this thesis. Highlighted boxes indicate parts of the search model covered in this thesis. The arrows shows both the flow of the information and dependency of chapters.

discuss the distribution representations and propose to use particle-based representation where each particle is modelled using an agent model of the LP movement.

We start describing the design of the agent model and develop a local agent model of LP movement with limited behaviour and local interaction capabilities in Chapter 4. We describe the various parts that comprise the agent model and then detail the design of the model in terms of local interactions only.

We build on this local agent model in Chapter 5 and introduce global interactions. Here, we demonstrate how considering behaviours like re-orientation strategies which are dependent on perception of environment both at local and global levels improve the performance of the agent model, allowing it to produce realistic trails. We also model state-dependent behaviours, such as fatigue into the agent model and illustrate through simple examples how this could be significant in the estimation of the LP.

The limitation of the agent model is the number of parameters used to encode various behaviours. Since all these parameters need to be defined with nominal values, we need to calibrate them using real world observations. This task is performed in Chapter 6. We demonstrate that after the calibration process, the agent is able to better model LP movement, and as a result generate more refined distribution.

Having calibrated the agent model parameters, we present the update model (in search phase) in Chapter 7. We model and demonstrate how UAV observations of evidence and land cover along with observations of the LP can be used to update the initial distribution. We demonstrate that by using the three different observation types in the update of the distribution, we can infer the location of the LP in much shorter times compared to current grid-based searches where only observation of LP is considered.

This far, we assume known searching platform and detected evidence feature locations. However, this is not the case in reality. To model uncertainty in the position of the detected evidence features from a flying platform and also in the position of platform itself, we present details of a localisation method in Chapter 8.

Having presented the details of the localisation method, we bring the complete search model together in the final experiment of the chapter. We show that even with uncertain location information, our search model is able to out-perform grid-based search model.

Finally in chapter 9, we present summary of the work done in this thesis, and possible future work that could follow from this research.

Chapter 2

Probabilistic Approach to Wilderness Search and Rescue

2.1 Introduction

This chapter describes the probabilistic formulation of a WiSAR operation. We begin by presenting the timeline of a typical WiSAR operation, describing each of the stages in Section 2.2. A probabilistic description of the search process is outlined in Section 2.3. This shows the relationship between initial distribution generation, update, and search path planning to determine where the UAV should travel next. Since it is a major focus of the thesis, Sections 2.4 and 2.5 describes the algorithms used to generate the initial distribution and, in particular, a grid-based diffusion approach which is state-of-art in probabilistic search.

We then analyse the effectiveness of this approach. Section 2.6 carries out two simulation experiments which show the process of initial distribution generation and search. Section 2.7 then critically discusses the results and argues that a more powerful initial distribution generation and search models are required. Finally, we provide a summary of the chapter in section 2.8.

Key terms used in this chapter

- **Initial distribution**: The distribution over the LP whereabouts before search operation is initiated at time-step $k = 0$.
- **Posterior distribution**: This refers to the updated distribution over the LP location at time-step $k > 1$ conditioned on all the measurements from time-step $k = 0$ to k.

2.2 Timeline of Wilderness Search and Rescue Operation

We base our description on the field study by Goodrich in [12] and study of real search cases in [3]. The timeline[1] of a typical mission is shown in Figure 2.1. Suppose a person is hiking

[1] Although it is usually the custom to use a single time subscript in modelling a problem, in this research however, to make the separation of search tasks and times they take place, clear, we use unique subscript for each.

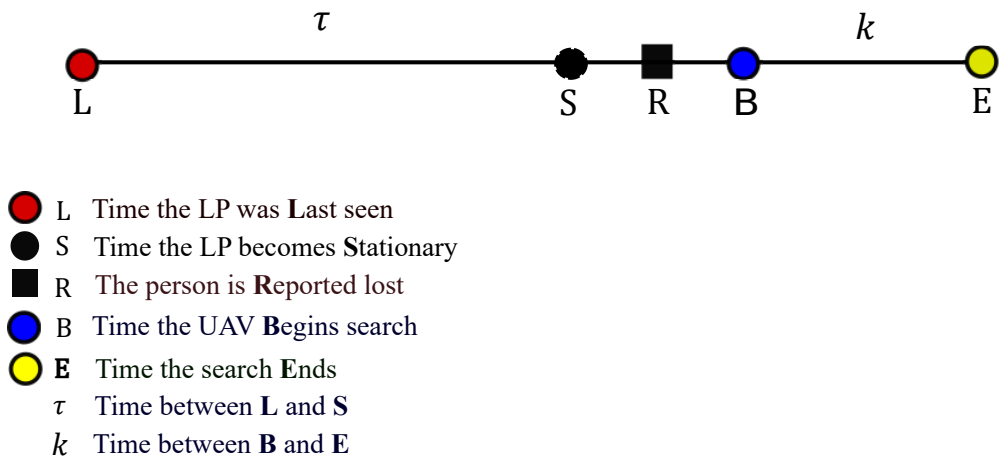


Figure 2.1: The timeline of a search scenario.

with his[2] friends. At time L, he heads off alone agreeing to meet his friends at a meeting point. However, at time S, he gets injured and fails to show up at the prearranged meeting point. After waiting for a while, his friends report him lost at time R. Having received the LP report, the WiSAR operation begins.

2.2.1 Initiation of Wilderness Search and Rescue

Upon receiving the LP report, the responders identify two things. The first is to establish a profile of the LP P. This includes possible places the person might intend to go, familiarity with terrain, age and the reason for the LP's visit to the area (hunting, hiking, etc). The second is to establish the Point Last Seen (PLS), which is the last reliable reported location of the LP.

However, the information provided can be incomplete. Therefore, a further information gathering phase is carried out.

2.2.2 Information Gathering Phase

In the information gathering phase, additional information is collected to: (a) compile an improved profile of the LP, (b) improve the PLS and (c) construct a representation of the environment. The additional information that can be compiled into the profile includes:

- *Physical description*: The LP's physical status: height, physical build (athletic or not) and any disabilities (especially those which affect walking). Knowing this information has significant impact on the search operation. People with different physical capabilities tackle the same type of terrain differently. For example, a person with fitness issues typically travels a shorter distance, so the search area is smaller.

- *Mental Health*: This is another important factor used in determining the LP's preferences. For example, for a despondent person, the search would be mostly performed in

[2]For simplicity and consistency we will use the preposition "him" and pronoun "he" in this research.

places where the person can hide. In contrast, for a hiker, the search would be mostly concentrated on parts of the search area that offer least resistance to movement and around linear features.

- *Possible intentions or goals*: Likely places the LP might aim to move towards. Most LPs typically start towards an end goal but get lost en route.

- *Details of clothing*: What the LP was wearing and carrying when last seen. Knowing the types of clothing a LP is wearing can help determine the importance of the detected evidence features later on in the search process.

A very important source of information is the behavioural statistics based on a compiled database of search cases [3]. The International Search and Rescue Incident Database (ISRID) has compiled a database of 50,000 search cases, 689 of which originate in the UK [32]. The analysis of these cases provide a wealth of information. For example, hunters frequently travel uphill when they are lost (in order to reorient) and hikers tend to follow linear features, such as ridge lines and trails, when they become disoriented. This book has extensively been referred to in search and rescue literature [33–40], and used by search and rescue teams in different search operations across the world as guidance. Other studies such as those in [36, 40, 41] complement [3], but provide no new information.

To improve on the reported PLS and, if possible the direction of travel, additional information includes things like sighting of the LP by other witnesses, and place where the LP's car or belongings which have been found are gathered.

Parallel to gathering information about the LP and the PLS, the WiSAR team compiles a model of the environment incorporating information pertaining to the land cover, weather, known trails and roads within the search area. While land cover type and linear features like trails constrains LP's movements, weather effects the LP's ability to lay down trails, the route taken and the visibility of the search area. For example fog effects the movement of the LP due to restricted visibility of the search area. Moreover, variability in lighting conditions make the process of land cover classification and evidence detection quite challenging.

2.2.3 Search Planning and Initial Distribution Generation Phase

In this phase, the incident commander develops a search plan. This incorporates two things: specification of the search area and the initial distribution generation. The search region $\mathcal{A}$ is specified based on information such as:

- The Point Last Seen (PLS).

- The time between L and R – the time the LP is assumed to have been mobile after PLS.
- The LP profile.
- Incident commander's knowledge and experience of WiSAR cases.
- Search area information in the form of land cover data sets.

Then an initial distribution is generated, which is used by the incident commander to prioritise regions within the search area and assign search resources accordingly.

To compute the initial distribution, the search team identifies the likely trails taken by the LP given the search area features. The trails are then prioritised using the LP's profile P. The initial distribution is then constructed by considering the aggregate weight of trails in each region. Using this initial distribution, the search for the LP begins.

Both information gathering and initial distribution generation phases happen between times R and B.

2.2.4 Search Phase

The search phase begins at time B by initiating the ground search operation with three overall objectives:

1. Maintain safety of searchers.
2. Locate the LP.
3. Rescue or recover the LP.

Teams of searchers follow one of the search paradigms listed in Appendix B.1 depending on factors like search environment type, time constraints and number of searchers.

During the search, the search teams look for and report any evidence found to other searchers and the incident commander, who monitors the activities of the search teams, including their locations. The incident commander updates the initial distribution with this new information. Thus, in the search phase, the initial distribution is recursively updated with new observations, which could be both positive (evidence has been detected) or negative (evidence has *not* been detected). The resultant updated distribution is called the *posterior distribution*.

Ideally, the posterior distribution peaks at a location in the search area where the LP is located ending the search mission at time E. However the search may also end if the search continues long enough without success or if constraints such as bad weather halts the search operation.

All of this done manually at present. However, as explained in the Chapter 1, self-piloted UAVs have the potential to transform these operations. As a first step, we need to reformulate the problem mathematically.

2.3 Probabilistic Search

There has been much interest in developing methods and algorithms to incorporate UAVs into search and rescue missions [17–20, 42–44].

The issue is that the whole problem is characterised by uncertainty – we do not know where the LP is, we do not quite know where the platform might be, and we have sensors which are noisy. Therefore, we use probability. A conventional way to formulate the search phase is to use the Bayesian framework [16, 19, 20, 45–48]. The primary reasons for this is Bayesian framework's applicability to general probability density functions, as well as its inherent recursive formulation.

2.3.1 Probabilistic Search Representation

A typical search problem is characterised by a large search area and sparse observations. Therefore, grid-based methods are widely used [16,18,19,44]. An example of this approach is shown in Figure 2.2. The search area defined by $\mathcal{A}$ is decomposed into $|\mathcal{A}| = M$ cells, where a_i is the i^{th} cell.

The distribution over the target's whereabouts is the equivalent of computing the probability that the target can be found in a given cell. As a result, the problem of search is posed in terms of minimising the entropy in the location of a target over a discretised/decomposed representation of the search area.

This representation suffers from two main issues: first, the computational cost scales linearly with the number of cells in the grid; secondly, its impracticality to large uncertain environments due to fixed grid size, which defines the localisation accuracy. However, because of its ease of handling and interpretation of data, specifically when dealing with highly non Gaussian distribution, and also its ability to integrate several sensors in the same framework taking their inherent uncertainty into account, this representation is still most efficient and widely used in search literature.

For example, the work in [17] uses a grid-based system to perform a Bayesian search for a target using a single UAV. Similarly, work presented in [16,46,47] have successfully employed the grid-based method in a variety of coordinated multi-UAV search problems, taking advantage of the method's ability to maintain representation for all states in the target space, regardless of the probability density. This is later extended by [48] to successfully simulate the coordinated

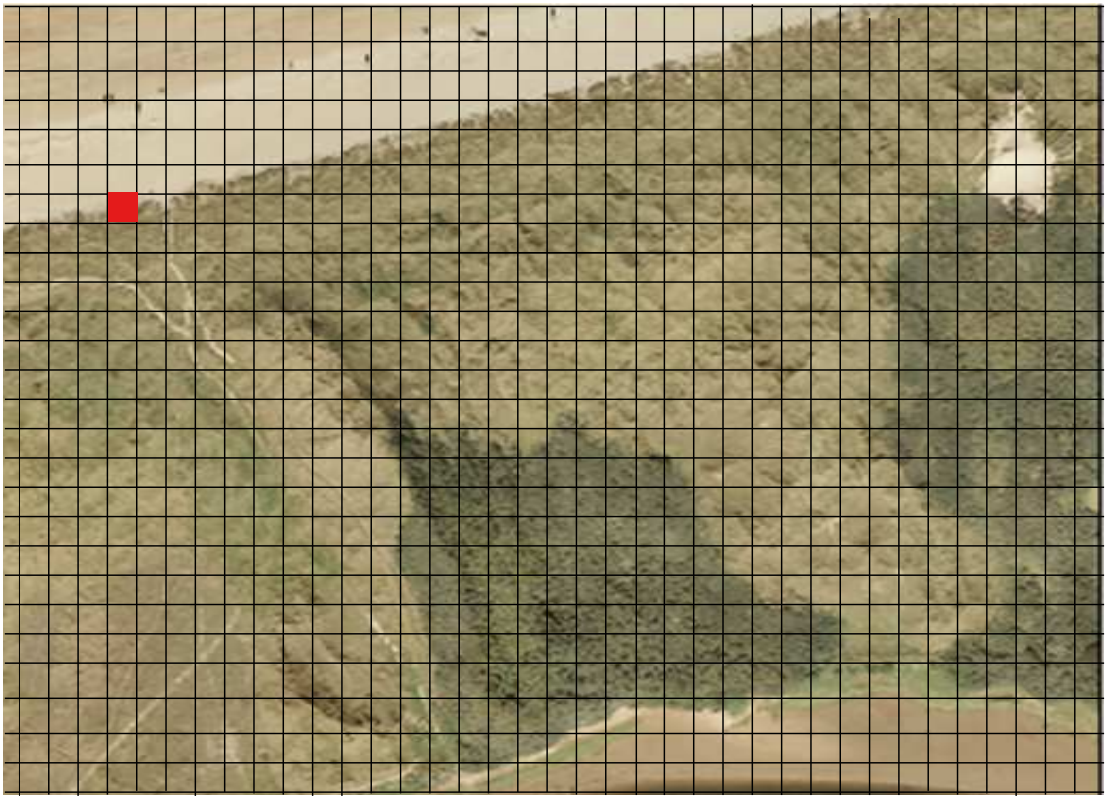


Figure 2.2: Illustration of using grids to represent the search area. The red square represents the last known point of the LP also called Point Last Seen (PLS).

search for multiple targets.

2.3.2 Problem Setup

Let $\mathbf{l}_k = [x \quad y]_k^T$ model the LP state at time k, where x and y are the Cartesian coordinates of location in 2D. Also, let Z_k^p represent the history of the observations of the LP (the super script p) up to and including time k, where $Z_k^p = \{z_1^p, \cdots, z_k^p\} = \{z_k^p, Z_{k-1}^p\}$, and $z_k^p \in \{0, 1\}$ a binary random variable. Search for a target is modelled by computing

$$p\left(\mathbf{l}_k | Z_k^p, P\right), \tag{2.1}$$

where P is the profile of the LP. This formulation can be solved using $p\left(\mathbf{l}_B | P\right)$ as the initial distribution at time B – a distribution over the LP whereabouts at the start of of search operation (time-step $k = 1$) when no observations have been made. Therefore, we need to model the initial distribution first and then compute (2.1).

2.3.3 Initial Distribution Generation

The initial distribution can be non-uniform. Some areas will have high probability of containing the LP, others a very low probability. As a result, the search area can be prioritised and the

UAV path can be computed to maximise the probability of the LP detection, thus efficiently distributing search resources [49].

To generate the initial distribution, we need to model the movement of the LP. Assuming first order Markov chain, the joint distribution along the entire path can be computed from

$$p\left(\mathbf{l}_{L:S}|P\right) = p\left(\mathbf{l}_S|\mathbf{l}_{S-1}, P\right) p\left(\mathbf{l}_{S-1}|\mathbf{l}_{S-2}, P\right) \cdots p\left(\mathbf{l}_L|P\right), \tag{2.2}$$

where $p\left(\mathbf{l}_L|P\right)$ is the distribution of the LP at the PLS, $\mathbf{l}_S$ corresponds to the end location of the LP and $p\left(\mathbf{l}_S|\mathbf{l}_{S-1}, P\right)$ models the LP's motion. Since we are interested in the current estimate of the LP location, we use the discrete analog to Chapman-Kolmogorov equation to model the propagation of the probability distribution [19]

$$p^*\left(\mathbf{l}_\tau = a_i|P\right) = \sum_{j=|\mathcal{A}|} p^{(i,j)} p\left(\mathbf{l}_{\tau-1} = a_j|P\right), \tag{2.3}$$

where τ is a time-step between L and S, $p^{(i,j)} = p\left(\mathbf{l}_\tau = a_i|\mathbf{l}_{\tau-1} = a_j, P\right)$ is the discrete state transition probability of the LP moving to cell i at time k, given that he was in cell j at time $k-1$, and $p\left(\mathbf{l}_{\tau-1} = a_j|P\right)$ is the probability that the LP was in a_j in the previous time-step $\tau - 1$. This has the effect of increasing the uncertainty in the LP's location in the search area. It reflects the discrete structure of the search area and captures the uncertainty in the target motion. The term $p^*\left(\mathbf{l}_\tau = a_i|P\right)$ is the probability that the LP will be in cell a_i at time-step τ. To initiate the search phase $p\left(\mathbf{l}_B = a_i|P\right) = p^*\left(\mathbf{l}_S = a_i|P\right)$ is used as initial distribution.

2.3.4 Discrete Bayesian Update Formulation

The update is carried out using Bayes' rule,

$$p\left(\mathbf{l}_k = a_i|Z_k^p, P\right) = \eta \cdot p\left(z_k^p|\mathbf{l}_k = a_i\right) p\left(\mathbf{l}_k = a_i|Z_{k-1}^p, P\right), \tag{2.4}$$

where $p\left(z_k^p|\mathbf{l}_k = a_i\right)$ is the likelihood of the LP observation, $p\left(\mathbf{l}_k = a_i|Z_{k-1}^p, P\right)$ is the probability of the LP being in a_i given detection measurements up to time-step $k-1$ and η is the normalising constant.

Measurements are taken in the presence of noise. As a result, the detector may incorrectly register the presence or absence of the LP within a cell. Therefore, we need to construct a detection model or likelihood model $p\left(z_k^p|\mathbf{l}_k\right)$ for imperfect detection

2.3.5 Target Detection

The presence (1) or absence (0) of the detection of the LP in the cell can be modelled by

$$p\left(z_k^p|\mathbf{l}_k\right) = \begin{cases} p\left(z_k^p = 0|\mathbf{l}_k = a_i\right) & = \beta^{(a_i)} \\ p\left(z_k^p = 1|\mathbf{l}_k = a_i\right) & = 1 - \beta^{(a_i)} \\ p\left(z_k^p = 0|\mathbf{l}_k \neq a_i\right) & = 1 - \alpha^{(a_i)} \\ p\left(z_k^p = 1|\mathbf{l}_k \neq a_i\right) & = \alpha^{(a_i)} \end{cases}, \tag{2.5}$$

where $p\left(z_k^p = 1|\mathbf{l}_k \neq a_i\right)$ models *false alarm* and $p\left(z_k^p = 0|\mathbf{l}_k = a_i\right)$ models *missed detection* with probabilities $\alpha^{(a_i)}$ and $\beta^{(a_i)}$ respectively.

These error probabilities characterise the noise characteristics of a sensor. For a given sensor both $\alpha^{(a_i)}$ and $\beta^{(a_i)}$ can be determined experimentally or from sensor specifications [18].

Sensors can only measure the state of the environment immediately surrounding the UAV. To observe the environment, the UAV has to move from place to place. The problem of deciding where to go next is called search path planning.

2.3.6 Unmanned Aerial Vehicle Search Path Planning

To explore the search area, the UAV needs to know where to go next to search for the target. This is achieved using a path planning algorithm.

To simplify the problem, the search literature normally assumes that the platform has a perfectly known pose, with movement constraint of one cell per time-step and sensing capability of exactly the size of one cell.

Many algorithms have been used for path planning such as a Voronoi Diagram with Eppstein's k-best paths algorithm [50], A* [51, 52], and Probabilistic Road-maps [53, 54]. These methods, however, focus on obstacle avoidance.

For search path planning, a common assumption is that the search platform i.e. the UAV is flying at a high enough altitude that it avoids obstacles. Therefore, the main objective is to locate the target. There are two categories of path planning methods normally used for this: methods that do not consider the spatial distribution of the target, and methods that do. Examples of the former category include random walk and sweep search. As its name implies, random walk search occurs when the UAV chooses, at random, the next cell to fly to. In sweep search, the search platform systematically visits each cell in order, using something like a lawnmower-like pattern [20]. Although these algorithms are easy to implement, they can lead to an inefficient search plan. Because they do not exploit information about the spatial distribution of the target,

the UAV can spend a significant amount of time visiting the environment where the LP cannot be found.

Using the spatial distribution of the target, an optimal path planning algorithm would include computing search path using the full spatial distribution, along which the objective function, which in our case is the detection of the target, is maximised. However, achieving the solution is often computationally extremely expensive [20, 55–57], and sub-optimal algorithms have to be used.

The solutions proposed vary from simple search strategies to complex path planning methods. For example, [11] and [20], propose five different search strategies:

- *Greedy search:* This is performed by moving to the immediate neighbouring cell with the highest probability of finding the target.
- *Offset contour search:* In contrast to greedy search, rather than seeking out probability differences, a given area is searched in a highly systematic fashion following a set of probability map contours - offset paths spaced d apart.
- *Composite search:* This search is executed by switching between the greedy search and offset contour search at different times. In general, the greedy search is used unless a certain degree of uniformity is present, and then the contour search is used.
- *Saccadic Search:* This search strategy focuses the search on the cell containing the maximal probability at every time-step executing jumps from peak to peak. The main difference compared to greedy search is that the cells containing the peaks can be located at a distance from one another.
- *Drosophila-inspired Search:* This search strategy provides an alternative to Saccadic Search, which is not constrained by the search platform's dynamics. It identifies the cell with the highest probability. Then the next cell to observe is determined by selecting a cell that lies within a certain distance (constrained by searcher dynamics) along the path to the identified cell. Once the cell with highest probability is reached, next peak cell (next cell with highest probability) is selected.

While the aforementioned are quite simple path planning methods, on the other end of spectrum exists some more complex ones. For example, in a series of papers [16, 42], Bourgault et al. propose path planning methods that in addition to distribution over the target whereabouts, to maximise the detection of the target, consider restricted time using one or multiple UAVs and human systems. Similarly, in [58], Lin et al. propose algorithms based on local hill

climbing and evolutionary algorithms that use techniques such as "global warming effect" and path crossover/mutation using distribution over a target's whereabouts to plan the UAV search path.

The important point to note is that there are many types of path planning algorithms. However, a common feature is that they all depend upon knowledge of the spatial distribution of the target. The more faithfully this distribution represents the target's whereabouts, the better and more efficient the search will be in terms of the time to locate the LP. For this reason, in this research we do not consider a new sophisticated path planning method or using any of the sophisticated path planning methods mentioned. Instead, we are more concerned with generating a higher quality initial distribution.

There are many approaches for search based on look-ahead [16, 20, 42]. In contrast to greedy search strategies such as local hill climbing, in this approach, the UAV bases its control decision not on the value of the belief of the LP's presence in the neighbouring cell, but on the overall gain of moving into that cell.

This approach uses a *look-ahead* window of w cells, over which the optimisation of the path is performed. Also known in optimal control as *receding horizon control* [59], this approach is performed in two steps: first, a sequence of potential search paths $\Pi = \{\pi^1, \cdots, \pi^n\}$ are identified. A potential path consists of list of cells to visit $\pi = a_c, \ldots, a_e$, where a_c is the current cell and a_e is a cell with highest probability of containing the LP within the window. Then, the optimal search path π^*, which maximises the objective function over the horizon window is identified by

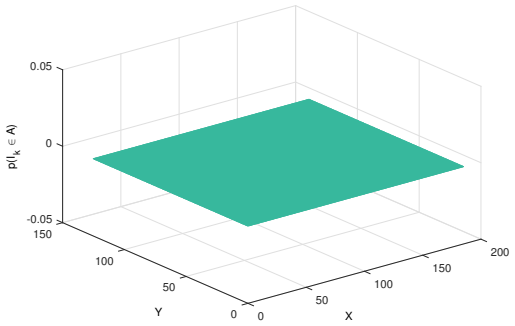
$$\pi^* = \max_{\pi} f(\pi),$$

where $f(\pi)$ is some objective function. In our case the objective function is the sum of the probability values of cells along the potential path, $f(\pi) = \sum_j^e p\left(\mathbf{l}_k = a_j | Z_k^p, P\right)$.

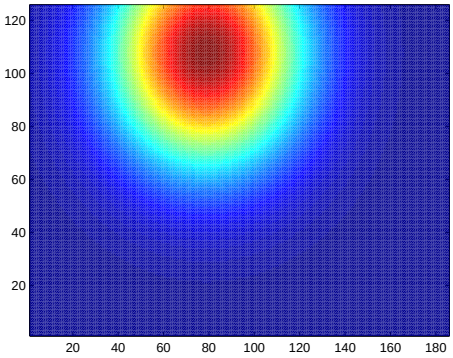
Then, first cell of this policy is selected to be the next cell to be visited in the next time-step. At the next n^{th} time-step, the UAV search path over the window is recomputed.

Since the computational cost of the method depends on the size of w, the planning window size w should be chosen to provide an approximation to the optimal solution while maintaining computational feasibility. In this research, we will use this path planning method with a window size of 5 and $n = 4$, unless otherwise stated.

Having presented the probabilistic search model, the initial distribution can be generated using various methods with varying levels of complexity. Here we present examples of these.



(a) Uniform initial distribution



(b) Gaussian initial distribution

Figure 2.3: Simple initial distribution types. The example Gaussian distribution is generated using $\mu = \begin{bmatrix} 79 & 108 \end{bmatrix}$ and $\sigma = \begin{bmatrix} 50 & 0 \\ 0 & 50 \end{bmatrix}$.

2.4 Simple Approaches to Initial Distribution Generation

The simple approaches mainly used to create the initial distribution are: *uniform*, *Gaussian* and *manual* distributions based.

As the name suggests, the uniform distribution, illustrated in Figure 2.3a, assumes that the target could be anywhere in the search area with the same probability [60]. It is used when no information is available about either the search area or the LP. On the other hand, the Gaussian distribution illustrated in Figure 2.3b uses the LP profile information such as speed and time elapsed since the LP was last seen [3, 16] to generate a more informative distribution. This gives a bell-shaped distribution over the LP's location centred on the PLS, which basically means higher probability of locating the LP nearer the centre. This form of initial distribution is used either when minimal information about the LP profile and the search area is available, or when the search team do not have access to more complex initial distribution generation tools.

In contrast to the previous two methods, manual distribution is specified by the search team manually by considering things like, places of interest the LP may want travel towards

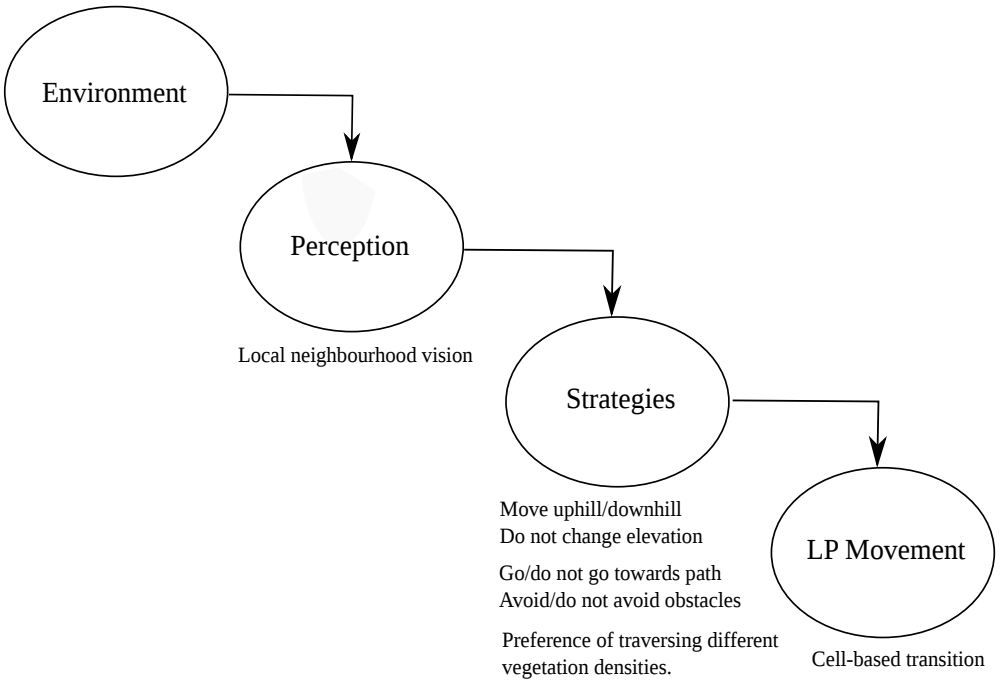


Figure 2.4: Design model of diffusion-based LP movement. The description bellow each level indicate the behaviours captured.

and environment configuration and its effect on the LP movements. An example of the latter can be found in [12].

2.5 Diffusion Approach to Initial Distribution Generation

A more sophisticated way to compute the initial distribution is to use a cell-based transition model. In this model, the LP's motion is computed using a transition matrix that models movement from one cell to its neighbouring cells considering both the difference in the feature classes between the current and neighbouring cells and the LP profile P. Examples of the cell-based transition model is given in [47, 61]. Here we will detail the approach most relevant to our problem domain proposed in [61].

2.5.1 Diffusion Model

This approach uses a subset of the LP profile information, together with the PLS and the time last seen to create a memory-less model of the movement of the LP. Profile information such as physical health, mental conditions and the nature of visit is used to categorise the LP and determine movement preferences. The Point Last Seen (PLS) and the time last seen are used together with physical health information to determine the size of the search area.

The diffusion approach models the LP's *local* interactions with the search environment using a four level design shown in Figure 2.4. The four levels in this design consists of:

- *Environment:* The representation of the search area based on data sets such as elevation

E, vegetation V and topography T.

- *Perception:* Observable part of the environment.
- *Movement:* The model by means of which movement decisions are executed within the environment.
- *Strategies:* Means by which lost people try to get re-oriented. Perception is mapped onto movement based upon strategies.

Let the event $\Lambda = \{P, E, T, V\}$ mean that the profile and data sets have particular values. Using this information, the term $p\left(\mathbf{l}_\tau = a_i|P\right)$ in (2.3) is replaced by $p\left(\mathbf{l}_\tau = a_i|\Lambda\right)$. With the model of LP dynamics defined, the initial estimate generation starts by initialising the distribution,

$$p^*(\mathbf{l}_L \in \mathcal{A}|\Lambda) = \begin{cases} p(\mathbf{l}_L = a_i|\Lambda) = 1 & \text{if } a_i = \text{PLS} \\ p(\mathbf{l}_L = a_i|\Lambda) = 0 & \text{if } a_i \neq \text{PLS} \end{cases}.$$

Then, the diffusion model in (2.3) is run Y times. In the τ^{th} iteration, the probability of occupancy of cell a_i is

$$p^*\left(\mathbf{l}_\tau = a_i|\Lambda\right) = \sum_{j=|\mathcal{A}|} p^{(i,j)} p\left(\mathbf{l}_{\tau-1} = a_j|\Lambda\right). \tag{2.6}$$

The transition probability $p^{(i,j)}$ depends on data sets representing the environment. Therefore, we consider the impact of these in detail.

2.5.2 Search Area Data Sets

There are three complementary aspects of the environment: the vegetation V, the topography T and the elevation E.

1. **Elevation data set E :** Elevation differences (slope) in the environment effect a person's speed, energy consumption and preference to move uphill or downhill. While speed and energy consumption are affected by actual elevation change, preference of moving uphill / downhill is influenced by the *perceived slope* of the environment. According to [62] humans naturally misjudge and overestimate the environment slope by high margins both verbally and visually. For example, humans tend to over estimate a 2° slope as 10° slope and 10° slope as 30° slope, which is further over estimated to a greater degree when suffering from fatigue [63].

   To model this perceived slope of the environment and its effect on the traversability, we

Table 2.1: Range of slope for each class of slope. The class extreme represents the uphill or downhill movement which becomes physically impossible for a person. The range of slope in this table correspond to the perception of slope by a hiker [62]. The hyphen $-$ means no range is specified in that specific class. Please refer to the text for a discussion of why uphill and downhill slopes ranges differ.

Class of slope	Category	Positive Slope Range	Negative Slope Range
S_1	Plain	$0^\circ < S \leq 5^\circ$	$0^\circ > S \geq -5^\circ$
S_2	Normal	$5^\circ < S \leq 10^\circ$	$-5^\circ > S \geq -10^\circ$
S_3	Down Hill	-	$-10^\circ > S > -30^\circ$
S_4	Uphill	$10^\circ < S < 40^\circ$	-
S_5	Extreme	$S > 40^\circ$	$S < -30^\circ$

have classified slope into five classes listed in Table 2.1

$$dom\left(S_{\left(E_{(a_i)},E_{(a_j)}\right)}\right) = \{S_1, S_2, S_3, S_4, S_5\}, \tag{2.7}$$

where $S_{\left(E_{(a_i)},E_{(a_j)}\right)}$ is the slope angle between cells a_i and a_j where $a_i \in \mathcal{A}$ and $a_j \in \mathcal{A}$. It is calculated from

$$S_{\left(E_{(a_i)},E_{(a_j)}\right)} = \tan^{-1}\left(\frac{E_{(a_i)} - E_{(a_j)}}{d(a_i, a_j)}\right), \tag{2.8}$$

where $d(a_i, a_j)$ is the Euclidean distance between the centres of the two cells in 2D,

$$d(a_i, a_j) = \sqrt{(x^{a_i} - x^{a_j})^2 + (y^{a_i} - y^{a_j})^2}, \tag{2.9}$$

where x^{a_i} and y^{a_i} are the coordinates of the centre of cell a_i and x^{a_j} and y^{a_j} are the coordinates of the centre of cell a_j. The asymmetry in uphill and downhill slope reflects a typical human's ability of going on steeper slopes uphill than downhill due to high motor control required to ensure speed is kept in check and to prevent falling and sliding down.

2. **Topography Classification data set T :** This data set classifies the search area in terms of its topography types, with

$$dom(T) = \{Obstacle, Water, Ground, Path\}, \tag{2.10}$$

which extends the topography classes in [61] by adding path and obstacle. Hikers cannot physically pass through some regions – such as obstacles (water or a high wall). Other regions such as paths might attract hikers, as illustrated in Figure 2.5.

Figure 2.5: Hikers following a path [64]. It shows why a path classification is needed.

3. **Vegetation Classification data set V:** This classifies the search area based on the density of vegetation using the categories in [61],

$$dom(V) = \{Sparse, Medium, Dense\}. \tag{2.11}$$

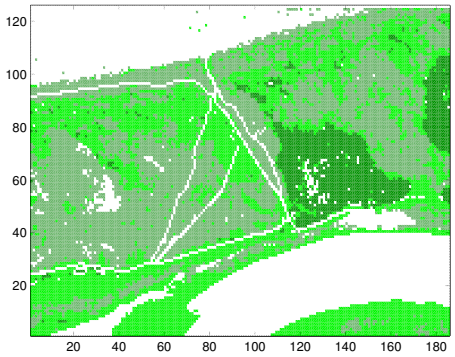
Figure 2.6 illustrates different kinds of vegetation. The difference in vegetation classes affect traversability. For example, hikers tend to keep clear of dense vegetation. They prefer traversing over sparse vegetation.

The elevation data set describes the search area in terms of its height above a fixed reference point and is based on a Digital Elevation Model (DEM). The last two data sets are acquired by classifying a high resolution image of the area. Both satellite images of the search area along with elevation data set can be acquired from sources such as [65–68] and the Met Office respectively. Figure 2.7 illustrates the satellite image of an area in Dartmoor UK along with elevation, vegetation and topography data sets used to represent the same area.

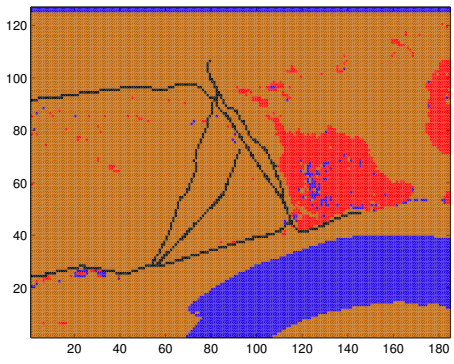
The three data sets are acquired in raster format. In this research, unless mentioned otherwise, each cell represents a $(5m)^2$ area. Details of the data set pre-processing and reason for the size of discretisation is given in Appendix A.

Figure 2.6: The general classes of vegetation considered.

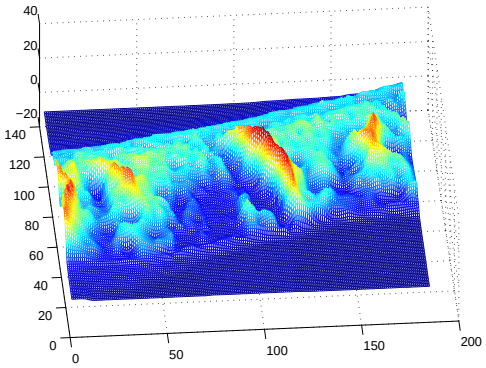
(a) Satellite image



(b) Vegetation data set



(c) Topography data set



(d) Elevation data set

Figure 2.7: The three data sets for one of the test areas in Sand Dunes in Northern Ireland.

2.5.3 Computing Transition Probabilities

In this section, we model the transition probability of the LP movements. The transition probability is determined by comparing the feature types in adjacent cells. This models the local interaction of the LP with the search area. Because we use three different data sets to represent the search area, we consider cell a_i with respect to each as illustrated in Figure A.2.

To compute $p\left(\mathbf{l}_\tau = a_i | \mathbf{l}_{\tau-1} = a_j, \Lambda\right)$ in (2.6) – a non standard constrained density function, inspired by MetropolisHastings algorithm [69], the state transition density function is assumed to be composed of joint distribution over two events: propose of a move and acceptance of the move. This joint density function is sampled by first proposing a move from a proposal distribution, which is based on sample's previous state. Then, the second step comprises of computing the acceptance probability of the candidate move which can take values 0 to 1 and is used to either reject or accept the candidate move with some probability. We call this a *propose–acceptance* approach.

In this approach, in a discretised environment, at time τ, the LP state is moved forward using a cell-based transition model, giving a proposed state of the LP. Then the event that this move is accepted (defined as A_τ) is computed. The probability of the acceptance is governed by the move being consistent with constraints placed by the search area (represented by the data sets) and profile of the LP. Therefore, defining the proposed move at time τ as $\tilde{\mathbf{l}}_\tau$, $\mathbf{l}_\tau = \{\tilde{\mathbf{l}}_\tau, A_\tau\}$, specifically

$$p\left(\tilde{\mathbf{l}}_\tau = a_i, A_\tau | \mathbf{l}_{\tau-1} = a_j, \Lambda\right) = p\left(A_\tau | \tilde{\mathbf{l}}_\tau = a_i, \mathbf{l}_{\tau-1} = a_j, \Lambda\right) p\left(\tilde{\mathbf{l}}_\tau = a_i | \mathbf{l}_{\tau-1} = a_j, T, P\right), \quad (2.12)$$

where the first term on the right hand side assesses the move proposed by the second term with respect to the search area data sets. It models the affects the physical constraints of the search area such the gradient of the terrain E, vegetation density V and the topography of the land cover T have on the LP's movements *locally* [3] [61]. Defining the neighbourhood cells as $N(.)$,

[3] This complex way of modelling $p\left(\mathbf{l}_\tau = a_i | \mathbf{l}_{\tau-1} = a_j, \Lambda\right)$ as propose–acceptance approach will allow us compare the model with a more detailed model later in the thesis

which include the current cell, the motion of the LP can be simply computed using

$$p\left(\tilde{\mathbf{l}}_\tau = a_i | \mathbf{l}_{\tau-1} = a_j, T, P\right) = \begin{cases} 1/|N(a_i)| & \text{if } a_j \in N(a_i) \text{ and } T(a_i) \neq Obstacle \\ 0 & \text{if } T(a_i) = Obstacle \\ 0 & \text{if } a_j \notin N(a_i) \end{cases}, \tag{2.13}$$

where $T(a_i)$ is the topography class of a_i.

To model the acceptance probability of the move, we need to consider how the different aspects of the search area affect the LP's choice. Assuming the effect of each data set is independent, the term $p\left(A_\tau | \tilde{\mathbf{l}}_\tau = a_i, \mathbf{l}_{\tau-1} = a_j, \Lambda\right)$ can be decomposed into the product of three terms, each one describing the impact of a different data set,

$$\begin{aligned} p\left(A_\tau | \tilde{\mathbf{l}}_\tau = a_i, \mathbf{l}_{\tau-1} = a_j, \Lambda\right) &= p\left(A_\tau^{(E)} | \tilde{\mathbf{l}}_\tau = a_i, \mathbf{l}_{\tau-1} = a_j, E, P\right) \\ &\times p\left(A_\tau^{(V)} | \tilde{\mathbf{l}}_\tau = a_i, \mathbf{l}_{\tau-1} = a_j, V, P\right) p\left(A_\tau^{(T)} | \tilde{\mathbf{l}}_\tau = a_i, \mathbf{l}_{\tau-1} = a_j, T, P\right), \end{aligned} \tag{2.14}$$

where $A_\tau^{(Q)}$ is the acceptance event of the move with respect to the classes of the data set specified by Q. Therefore, the terms on the right model the effects of terrain elevation change, vegetation and topography respectively.

2.5.3.1 Computing A_τ

When a move is proposed using (2.13), the effects of the search environment on computing the acceptance probability (2.14) for the proposed move is modelled using transition probability values. These transition probability values are acquired from the transition probability matrices $\mathbf{M}^V$, $\mathbf{M}^T$ and $\mathbf{M}^E$.

For example, with vegetation data set V classified into three classes in Section 2.5.2, a 3×3 transition matrix is used to model the transition of the LP from any one of the vegetation classes to any other vegetation class,

$$\mathbf{M}^V = \begin{bmatrix} V_{(1,1)} & V_{(1,2)} & V_{(1,3)} \\ V_{(2,1)} & V_{(2,2)} & V_{(2,3)} \\ V_{(3,1)} & V_{(3,2)} & V_{(3,3)} \end{bmatrix}. \tag{2.15}$$

The rows and columns in (2.15) are indexed by numbers representing the different classes of vegetation data set detailed in Table 2.2. In subscript (a, b), a represents the feature class at current location and b the feature class at destination location. For example, $V_{(1,1)}$ corresponds to the transition of the LP from vegetation class *sparse* to *sparse*. The same logic is applied to

Table 2.2: Description of subscripts used in transition matrices.

Data set	Subscripts used and corresponding class of data set
V	1 = Sparse 2 = Medium 3 = Dense vegetation.
T	1 = Obstacle 2 = Water 3 = Ground 4 = Path
E	1 = Plain 2 = Normal 3 = Downhill 4 =Uphill 5= Extreme

topography data set T in (2.10),

$$\mathbf{M}^T = \begin{bmatrix} T_{(1,1)} & T_{(1,2)} & T_{(1,3)} & T_{(1,4)} \\ T_{(2,1)} & T_{(2,2)} & T_{(2,3)} & T_{(2,4)} \\ T_{(3,1)} & T_{(3,2)} & T_{(3,3)} & T_{(3,4)} \\ T_{(4,1)} & T_{(4,2)} & T_{(4,3)} & T_{(4,4)} \end{bmatrix}, \tag{2.16}$$

where for example the term $T_{(2,4)}$ corresponds to the transition of the LP from topography class *water* to topography class *path*.

However for elevation, because we consider only five slope ranges (2.7), a 1×5 matrix is used to model this,

$$\mathbf{M}^E = \begin{bmatrix} S_1 & S_2 & S_3 & S_4 & S_5 \end{bmatrix}. \tag{2.17}$$

The single subscript used in (2.17) identifies the range of slope class corresponding to the elevation difference between the agent's current position and the next proposed position detailed in Table 2.1. Considering the example in Figure 2.8, where the green cell represents elevation at the LP's current location, the red cell represents elevation at the proposed location, and the arrow represents direction of the move. Using (2.8), $S_{\left(E_{(a_i)},E_{(a_i)}\right)} = 21.7^\circ$. Comparing this to slope classes in Table 2.1, we can determine that the move corresponds to slope class uphill i.e. S_4.

To acquire values for (2.15), (2.16) and (2.17), subjective expert opinion is often used to specify the belief on how the LP behaves with respect to different transitions [61].

Ambiguities and errors from subjective expert opinion are modelled by placing probability distributions over the values of each term in these matrices. Although these should be modelled by a probability distribution such as a Dirichlet process, the search literature often simplifies this and assumes a Gaussian [61].

This is done by setting mean μ and standard deviation σ for each of the possible terms in (2.15), (2.16) and (2.17). This is given in Table 2.3. For example, considering the vegetation data set of the search area, a lost hiker would mostly stay in sparse vegetation. For this reason, the terms $\mu_{V_{(1,1)}}$, $\mu_{V_{(2,1)}}$ and $\mu_{V_{(3,1)}}$ are set to high values and $\mu_{V_{(1,3)}}$, $\mu_{V_{(2,3)}}$ and $\mu_{V_{(3,3)}}$ are

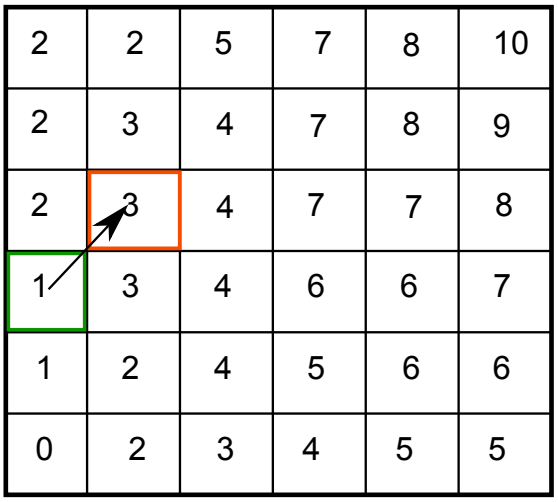


Figure 2.8: Example elevation data set. The green cell (current location of the LP) and red cell (next proposed location of the LP) using the elevation data set. The arrow indicates directionality.

Table 2.3: The data set transition matrices with parameters specified using Gaussian distributions. Although this should be done by experts in the field, for this thesis we have specified the parameters based on our knowledge of the search literature and the behaviour of the LP archetype. To model uncertainty in our knowledge, we have specified large standard deviation on each parameter. The mean values are assigned so that each row of the transition matrices sums to 1.

Parameter	Symbol	Value
$\mathbf{M}_{\mu}^{T}$	$\begin{bmatrix} \mu_{T_{(1,1)}} & \mu_{T_{(1,2)}} & \mu_{T_{(1,3)}} & \mu_{T_{(1,4)}} \\ \mu_{T_{(2,1)}} & \mu_{T_{(2,2)}} & \mu_{T_{(2,3)}} & \mu_{T_{(2,4)}} \\ \mu_{T_{(3,1)}} & \mu_{T_{(3,2)}} & \mu_{T_{(3,3)}} & \mu_{T_{(3,4)}} \\ \mu_{T_{(4,1)}} & \mu_{T_{(4,2)}} & \mu_{T_{(4,3)}} & \mu_{T_{(4,4)}} \end{bmatrix}$	$\begin{bmatrix} 0.10 & 0.10 & 0.35 & 0.45 \\ 0.10 & 0.10 & 0.35 & 0.45 \\ 0.05 & 0.05 & 0.35 & 0.55 \\ 0.05 & 0.05 & 0.35 & 0.55 \end{bmatrix}$
$\mathbf{M}_{\sigma}^{T}$	$\begin{bmatrix} \sigma_{T_{(1,1)}} & \sigma_{T_{(1,2)}} & \sigma_{T_{(1,3)}} & \sigma_{T_{(1,4)}} \\ \sigma_{T_{(2,1)}} & \sigma_{T_{(2,2)}} & \sigma_{T_{(2,3)}} & \sigma_{T_{(2,4)}} \\ \sigma_{T_{(3,1)}} & \sigma_{T_{(3,2)}} & \sigma_{T_{(3,3)}} & \sigma_{T_{(3,4)}} \\ \sigma_{T_{(4,1)}} & \sigma_{T_{(4,2)}} & \sigma_{T_{(4,3)}} & \sigma_{T_{(4,4)}} \end{bmatrix}$	$\begin{bmatrix} 0.03 & 0.03 & 0.07 & 0.09 \\ 0.03 & 0.03 & 0.07 & 0.09 \\ 0.01 & 0.01 & 0.07 & 0.10 \\ 0.01 & 0.01 & 0.07 & 0.10 \end{bmatrix}$
$\mathbf{M}_{\mu}^{V}$	$\begin{bmatrix} \mu_{V_{(1,1)}} & \mu_{V_{(1,2)}} & \mu_{V_{(1,3)}} \\ \mu_{V_{(2,1)}} & \mu_{V_{(2,2)}} & \mu_{V_{(2,3)}} \\ \mu_{V_{(3,1)}} & \mu_{V_{(3,2)}} & \mu_{V_{(3,3)}} \end{bmatrix}$	$\begin{bmatrix} 0.45 & 0.35 & 0.2 \\ 0.45 & 0.35 & 0.2 \\ 0.50 & 0.40 & 0.1 \end{bmatrix}$
$\mathbf{M}_{\sigma}^{V}$	$\begin{bmatrix} \sigma_{V_{(1,1)}} & \sigma_{V_{(1,2)}} & \sigma_{V_{(1,3)}} \\ \sigma_{V_{(2,1)}} & \sigma_{V_{(2,2)}} & \sigma_{V_{(2,3)}} \\ \sigma_{V_{(3,1)}} & \sigma_{V_{(3,2)}} & \sigma_{V_{(3,3)}} \end{bmatrix}$	$\begin{bmatrix} 0.09 & 0.07 & 0.035 \\ 0.15 & 0.07 & 0.035 \\ 0.10 & 0.09 & 0.02 \end{bmatrix}$
$\mathbf{M}_{\mu}^{E}$	$\begin{bmatrix} \mu_{S_{(1)}} & \mu_{S_{(2)}} & \mu_{S_{(3)}} & \mu_{S_{(4)}} & \mu_{S_{(5)}} \end{bmatrix}$	$\begin{bmatrix} 0.35 & 0.3 & 0.149 & 0.20 & 0.001 \end{bmatrix}$
$\mathbf{M}_{\sigma}^{E}$	$\begin{bmatrix} \sigma_{S_{(1)}} & \sigma_{S_{(2)}} & \sigma_{S_{(3)}} & \sigma_{S_{(4)}} & \sigma_{S_{(5)}} \end{bmatrix}$	$\begin{bmatrix} 0.07 & 0.07 & 0.05 & 0.05 & 0.0001 \end{bmatrix}$

set to low values, as the latter correspond to the transition to dense vegetation. The inherent uncertainty in expert knowledge is modelled by high standard deviation values given for each of the transition terms.

However, with large uncertainties, there is high chance of getting negative probability values. To address this, we use truncated Gaussian distributions, which cannot have negative samples.

Once distribution over transition parameters are specified, the transition terms in (2.15), (2.16) and (2.17) are acquired by sampling from respective Gaussian distributions,

$$\mathbf{M}^V \sim \mathcal{G}(\mathbf{M}^V_\mu, \mathbf{M}^V_\sigma). \tag{2.18}$$

Similarly

$$\mathbf{M}^T \sim \mathcal{G}(\mathbf{M}^T_\mu, \mathbf{M}^T_\sigma). \tag{2.19}$$

$$\mathbf{M}^E \sim \mathcal{G}(\mathbf{M}^E_\mu, \mathbf{M}^E_\sigma). \tag{2.20}$$

The sampled transition matrices are normalised so that each row sums to 1. Later, when modelling the acceptance terms in (2.14), we acquire values from the transition matrices in (2.15), (2.16) and (2.17) accordingly,

$$p\left(A^{(V)}_\tau | \tilde{\mathbf{l}}_k = a_i, \mathbf{l}_{\tau-1} = a_j, V, P\right) = \mathbf{M}^V_{(V(a_i),V(a_j))}. \tag{2.21}$$

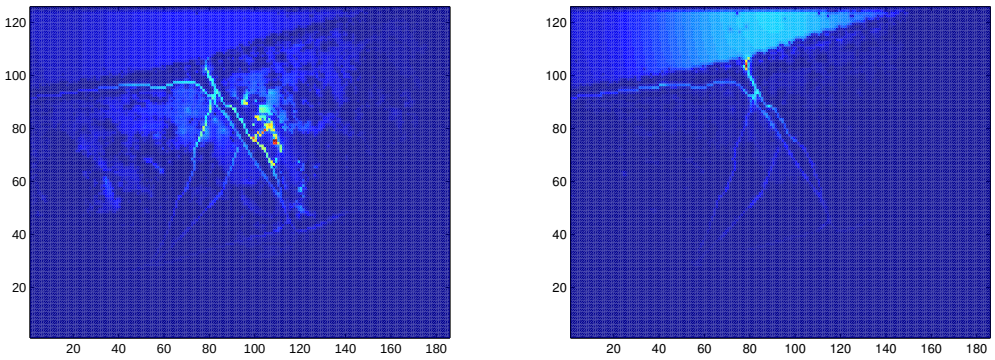
$$p\left(A^{(T)}_\tau | \tilde{\mathbf{l}}_k = a_i, \mathbf{l}_{\tau-1} = a_j, T, P\right) = \mathbf{M}^T_{(T(a_i),T(a_j))}. \tag{2.22}$$

$$p\left(A^{(E)}_\tau | \tilde{\mathbf{l}}_k = a_i, \mathbf{l}_{\tau-1} = a_j, E, P\right) = \mathbf{M}^E_{(E(a_i),E(a_j))}. \tag{2.23}$$

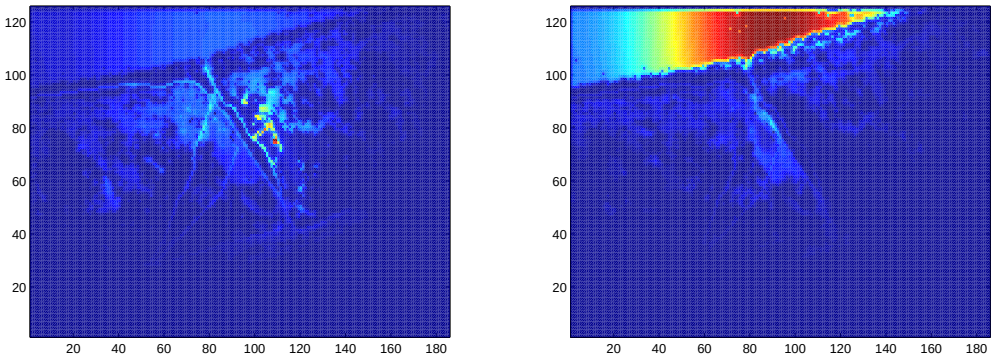
Assigning value to the terms in Table 2.3 plays a key role in modelling the LP's behaviour and interaction with the search area. This is illustrated in Figure 2.9[4]. The set of Figures 2.9a and 2.9b illustrate the generated distribution with uphill movement preference and downhill movement preference, which is achieved by changing $\mathbf{M}^E_\mu$ in Table 2.3 to $\begin{bmatrix} 0.15 & 0.15 & 0.2 & 0.4 & 0.1 \end{bmatrix}$ and $\begin{bmatrix} 0.15 & 0.15 & 0.4 & 0.2 & 0.1 \end{bmatrix}$ respectively, and then computing (2.18) to (2.23). As can be seen, the generated distribution in each case reflect the values set for $\mathbf{M}^E_\mu$. For example, when there is uphill movement preference Figure 2.9a, the generated distribution peaks at areas corresponding to the top of the hills. Similarly with down hill movement Figure 2.9b, the concentration of the distribution is along the shoreline. Since path preference is set to high values ($\mathbf{M}^T_\mu$ in Table 2.3), in each case paths have been distinctively identified.

The second set of Figures 2.9c and 2.9d illustrate the same elevation preferences when transition to path and staying on normal ground have same preference in $\mathbf{M}^T_\mu$ (Table 2.3),

[4]We detail the process of generating the initial distribution in the next section



(a) Uphill movement preference with path preference **(b)** Downhill movement preference with path preference



(c) Uphill movement with no path preference **(d)** Downhill movement with no path preference

Figure 2.9: Diffusion-based initial distribution for different elevation and topography transition preferences.

$$\mathbf{M}_{\mu}^{T} = \begin{bmatrix} 0.10 & 0.10 & 0.40 & 0.40 \\ 0.10 & 0.10 & 0.40 & 0.40 \\ 0.05 & 0.05 & 0.45 & 0.45 \\ 0.05 & 0.05 & 0.45 & 0.45 \end{bmatrix}. \tag{2.24}$$

Comparing the generated distribution with the previous set, we can see that, although the elevation related information stays similar, paths have vaguely been identified.

2.6 Implementation of Grid-Based Search

The implementation pseudo code of the diffusion-based initial distribution generation is given in Algorithm (1), which uses Algorithm (2) to compute the environment traversibility.

We will explain the initial distribution generation with the help of an experiment. Consider the example in Figure 2.10. The initial distribution generation takes as input the width w and height h of the discretised representation of the search area in terms of number of cells, which

Algorithm 1 Diffusion-based initial distribution generation

DiffusionBasedInitialDistributionGeneration()
INPUTS: Search area height and width in number of cells h, w respectively.
OUTPUT: $p(\mathbf{l}_B|\Lambda)$
Initialise State Vector $\mathbf{l}_L^D = a_i : a_i = \text{PLS}, |\mathbf{l}_L^D| = hw \times 1$
Initialise Environment transition matrix:
$\mathbf{A}_\tau = ComputeEnvironmentTraversibilityMatrix()$
for $\tau = L + 1 : S$ **do**
  if $\tau \% n = 0$ **then**
    $\mathbf{A}_\tau = ComputeEnvironmentTraversibilityMatrix()$
  end if
  Update the LP state $\mathbf{l}_\tau^D = \mathbf{A}_\tau \mathbf{l}_{\tau-1}^D$
end for

are 30 and 40 cells respectively in this example. Each cell representing $(25m)^2$.

Algorithm 2 Compute environment traversibility matrix

ComputeEnvironmentTraversibilityMatrix()
INPUTS: Λ, h, w
OUTPUT: $\mathbf{A}_\tau$
$\mathbf{M}^V \sim \mathcal{G}(\mathbf{M}_\mu^V, \mathbf{M}_\sigma^V)$
$\mathbf{M}^T \sim \mathcal{G}(\mathbf{M}_\mu^T, \mathbf{M}_\sigma^T)$
$\mathbf{M}^E \sim \mathcal{G}(\mathbf{M}_\mu^E, \mathbf{M}_\sigma^E)$
for $i = 1 : w$ **do**
  for $j = 1 : h$ **do**
    Let a_c represent the cell corresponding to coordinate i, j
    $Ldx = FindLinearIndex\,(N(a_c))$
    for $n = 1 : |Ldx|$ **do**
      Compute the move probability:
      $w_\tau^{(n)} = p\left(\tilde{\mathbf{l}}_\tau = a_{Ldx(n)}, A_\tau | \mathbf{l}_{\tau-1} = a_c, \Lambda\right)$
    end for
    Normalise the weights
    for $n = 1 : |Ldx|$ **do**
      Normalise $w_\tau^{(n)} = \frac{w_\tau^{(n)}}{\sum_i^{|Ldx|} w_\tau^{(i)}}$
    end for
    $ldx = LinearIndex(a_c)$
    for $m = 1 : |Ldx|$ **do**
      Update state transition matrix:
      $\mathbf{A}_\tau(Ldx(m), ldx) = w_\tau^{(m)}$
    end for
  end for
end for

With these inputs given, initial distribution generation starts by initialising the LP state. This is a state vector $\mathbf{l}_\tau^D$ of length of 1200×1, representing each cell in the search area. The state vector is initialised by setting it to 1 in the position that corresponds to the cell containing the PLS illustrated in Figure 2.10a and 2.10b. Then, the environment traversibility matrix is

computed using Algorithm (2), resulting in a traversibility matrix of size 1200×1200, computed using (2.12) - (2.23). Because the LP can only translate in one time-step to its immediate neighbours, this matrix is very sparse.

With these elementary steps completed, the process of initial distribution generation begins. The distribution over the LP location is recursively computed by

$$\mathbf{l}_{\tau}^{D} = \mathbf{A}_{\tau}\mathbf{l}_{\tau-1}^{D}.$$

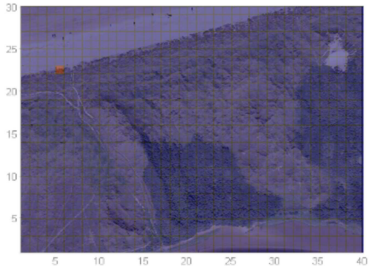
This is illustrated in Figure 2.10c to Figure 2.10h. With each iteration, the probability transitions out further and further into the environment. After 3000 time-steps (which corresponds to 165 minutes) Figure 2.10h, the spatial distribution is highly non-uniform: some areas have high probability of containing the LP, and others low.

Having generated the initial distribution, the search phase begins. The search is performed following the steps in Algorithm (3).

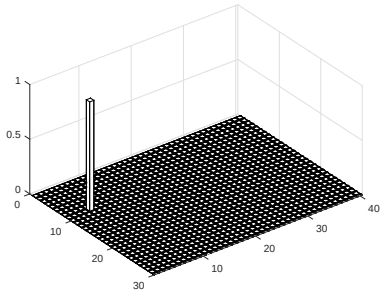
Algorithm 3 Grid-based search for a target

$\mathbf{GridBasedSearch}()$
INPUTS: $K, \alpha^{(a_i)}, \beta^{(a_i)}, \gamma, p\left(\mathbf{l}_B|\Lambda\right)$
Set $k = 1$
Initialise UAV state $\mathbf{x}_k = a_i : a_i = \text{PLS}$
while $Search$ **do**
  Get UAV Observation z_k^p = GetUAVObservation $(\mathbf{x}_k)$
  for $i = 1 : w$ **do**
    for $j = 1 : h$ **do**
      Set $a_m = \mathcal{A}_{(i,j)}$
      if Current cell(a_m) is observed **then**
        $p\left(\mathbf{l}_k = a_m|Z_k^p, P\right) = \eta \cdot p\left(z_k^p|\mathbf{l}_k = a_m\right) p\left(\mathbf{l}_k = a_m|Z_{k-1}^p, P\right)$
      else
        $p\left(\mathbf{l}_k \neq a_m|Z_k^p, P\right) = \eta \cdot p\left(z_k^p|\mathbf{l}_k \neq a_m\right) p\left(\mathbf{l}_k \neq a_m|Z_{k-1}^p, P\right)$
      end if
    end for
  end for
  Check search termination condition
  if $k > K \;\; || \;\; \max_{a_i \in \mathcal{A}} p(\mathbf{l}_k = a_i|Z_k^p, P) >= \gamma$ **then**
    $Search = false$
  end if
  $\mathbf{x}_{k+1}$ = UAVSearchPathPlanning($\mathbf{x}_k, p\left(\mathbf{l}_k|Z_k^p, P\right)$)
  Set $k = k + 1$
end while

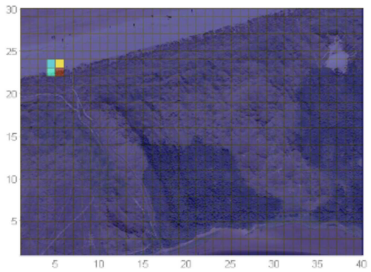
Defining K and γ to be the maximum search time allowed and detection confirmation threshold, and setting probabilities of false alarm ($\alpha^{(a_i)}$) and missed detection ($\beta^{(a_i)}$) to 0.2 and 0.3 respectively, the result of update is presented in Figure 2.11. As can be seen from



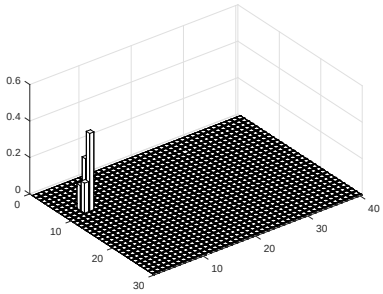
(a) $\tau = 1$



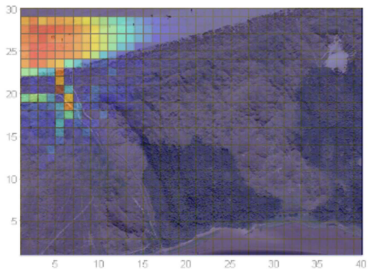
(b) $\tau = 1$



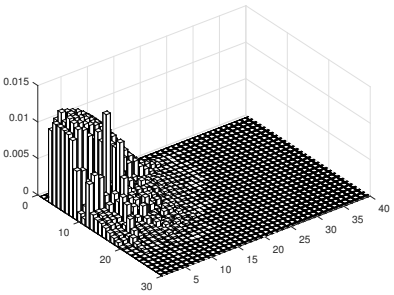
(c) $\tau = 2$



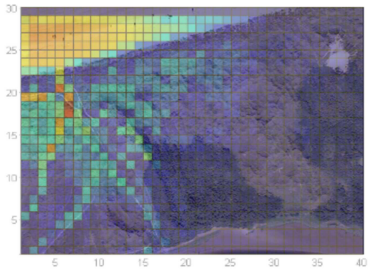
(d) $\tau = 2$



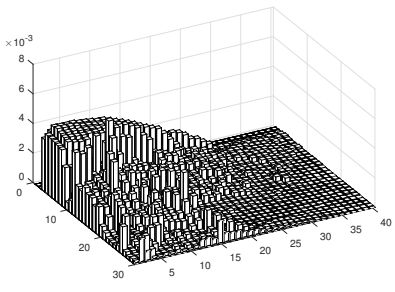
(e) $\tau = 440$



(f) $\tau = 440$



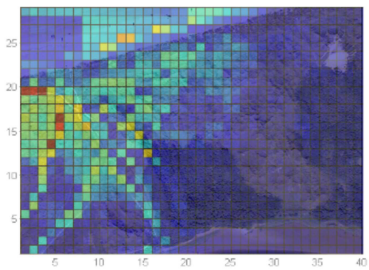
(g) $\tau = 3000$



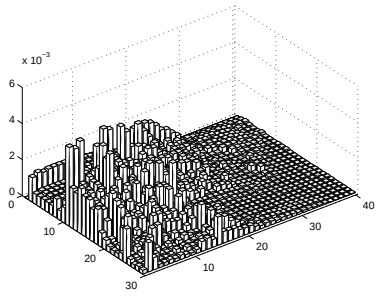
(h) $\tau = 3000$

Figure 2.10: An example of the initial distribution on the LP location at different time-steps generated using the diffusion model. Figures on the left column illustrate the generated distribution overlaid on the image of the search area. The intensity of cell colours in the overlaid distribution represent the probability that the LP is in the cell. High intensity representing higher probability values. Low intensity representing lower probability values. The Figures on the right column represent the same distributions using bar plots, where height of a bar represents the probability of the LP being in that cell.

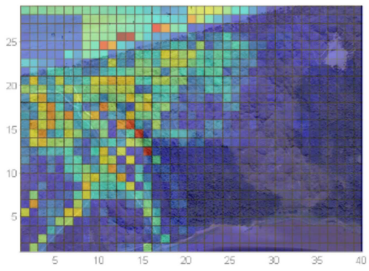
Figure 2.11a to Figure 2.11h, the distribution over the location of the LP is continually updated with observations of the search area using (2.4). Finally, at time-step $k = 175$ (Figure 2.11e and Figure 2.11f), upon positive observation of the LP, the distribution around the location of the LP peaks, and the distribution elsewhere goes down. Due to the imperfect sensor, it takes several observations of the LP in the cell to confirm the presence of the LP in the cell and refine the distribution to his end location at time-step $k = 180$, shown in Figure 2.11g and Figure 2.11h. Figure 2.12 illustrates the cells visited during the search.



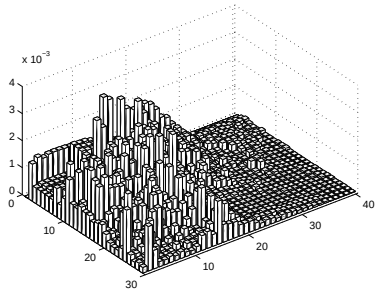
(a) $k = 120$



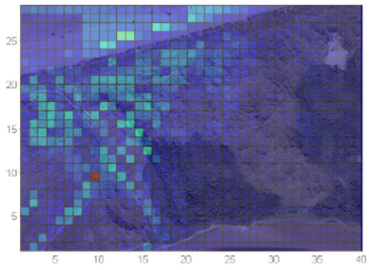
(b) $k = 120$



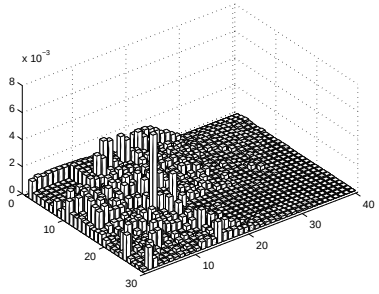
(c) $k = 173$



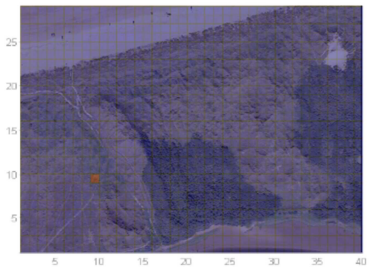
(d) $k = 173$



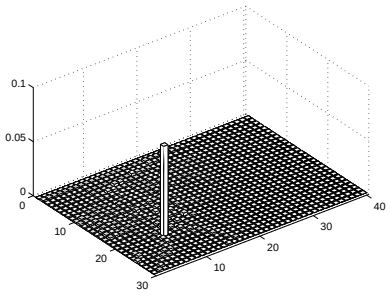
(e) $k = 175$



(f) $k = 175$



(g) $k = 180$



(h) $k = 180$

Figure 2.11: Example of grid-based updated distribution. Figures on the left column illustrate the posterior distribution overlaid on the image of the search area at different time-steps. Figures on the right represent the same distributions using bar charts to show in 3D the spatial evolution of the probability distribution. In the final Figure, the probability of the location of the target has been strongly placed in a single cell.

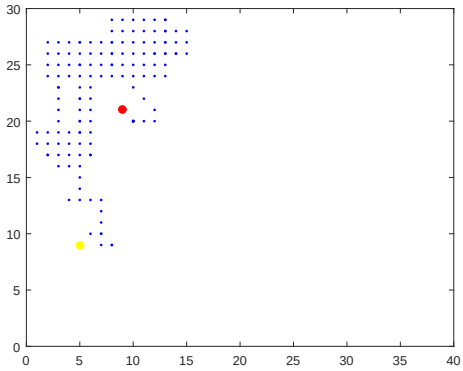


Figure 2.12: Cells visited during the search. Red and yellow dots marking the first and last cell visited.

2.7 Limitations of Conventional Grid-Based Search

Although a grid-based search approach is conceptually straightforward to implement, comparing it with the WiSAR operations described in Section 2.2 shows that there are some important limitations. We now discuss these.

2.7.1 During Initial Distribution Generation

Although the diffusion-based LP dynamics can model a number of environmental effects, such as the dependence on local slope, they do not model the fact that an LP is an intelligent and active entity whose internal state evolves over time. Furthermore, the actions of this entity are not influenced by just the immediately local neighbours, but can be influenced globally. Specifically:

- Way finding capability, which is highly dependent on the LP's perception of the environment [70], where perception consists of what the LP can see and what he can remember about the environment [71–73]. For example, if a LP has a clear line of sight over a long distance, they can set goals based on visible objects which are far away.

- Fatigue, which is a function of the trail taken and elevation changes in the environment. People cannot continue moving or traversing when they become fatigued [74]. This is illustrated using a simple synthetic example in Figure 2.13, where the blue trail moves a maximum distance before stopping and the red trail stops much shorter, as a result of going uphill and becoming fatigued[5].

- Tendency to walk in a straight line unless faced with an obstacle [75].

- Tendency to reduce total angle turned [76] when changing direction.

[5]Details of how this is implemented will be discussed later in this thesis.

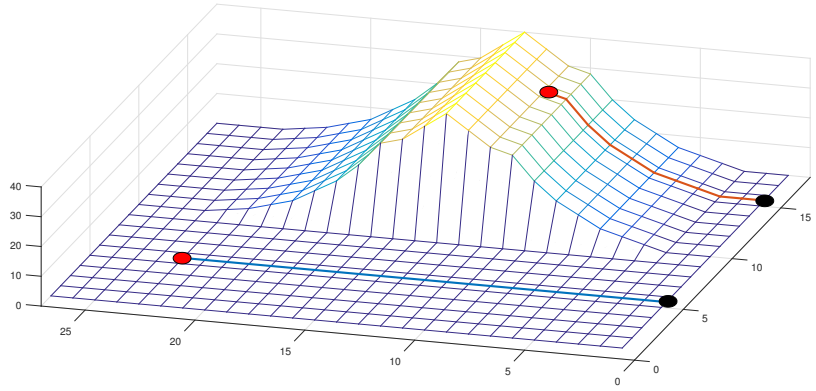


Figure 2.13: Example showing effect of elevation on movement. When a LP has to walk up a slope, they become tired more quickly and so do not travel as far. Black circles marking the start and red circles marking the end location for each agent trail.

- Switching between different strategies to re-orient themselves [3, 74]. One example is *direction travelling* [74], in which a person will walk up to a place of interest, ignoring other features and traversing over what would normally be considered hard to traverse regions.

Another important limitation is that it does not store information about the entire trail the LP undertook. As we discuss below, this means that, some kinds of information (which show that an LP could not have undertaken a particular route) cannot be used.

2.7.2 During Search Phase

The grid-based search method assumes that the only information that can be used is the presence or absence of the LP in a cell. As a result, it fails to exploit the fact that LPs tend to leave / drop evidence behind - very valuable sources of information, which search teams use to constrain the search area [10] reducing the search times and improving the survivability rate of the LP.

Although many of these evidence features, such as broken twigs and bruised vegetation, are probably too small to be automatically detected from the air, a small subset examples of which are illustrated in Figures 1.5 and 2.14a can provide with valuable information to help infer the Lost Person trail or intent.

Furthermore, it is assumed that the data sets used to compute the initial distribution are complete and up-to date. In reality, however, these data sets can be out of date. For example, the growth of vegetation might mean that a path which appeared on a map might no longer exist. Conversely, the movement of animals might have created a path which was not visible before, but which an LP might follow.

This means that the generated initial distribution could be a poor representation of the

(a) (b)

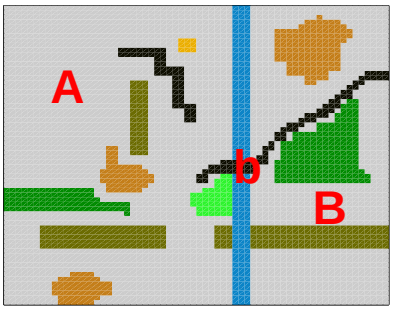
Figure 2.14: Some of the objects that can be found when searching for a LP. A piece of cloth, a food box and a cup captured by one of our UAVs in Dartmoor.

LP's whereabouts. For example, consider the scenario in Figure 2.15, where the search area $\mathcal{A}$ contains two regions A and C connected by the bridge b. The PLS is in region A. However, it turns out in reality that the bridge was damaged and has been impassible for some time. As a result, the LP must lie in region A.

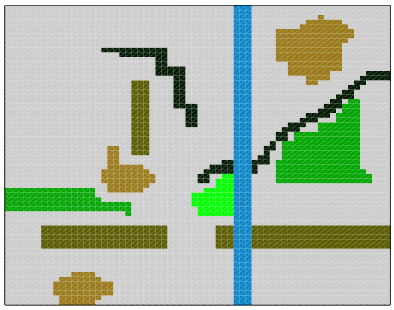
Having generated the initial distribution using the diffusion model initiated at PLS illustrated in Figure 2.15c. During the search phase, when it is determined that the bridge is broken Figure 2.15b, the probability of the LP being in region C should decrease to zero $p(\mathbf{l}_k \in C) = 0$ since the LP could not have crossed the bridge to get to region C. But as can be seen, the grid-based search method is not able to take advantage of this information, and is not able to refine the spatial distribution over the LP's location. This is because the trail or movement history of the LP has been ignored; therefore, it is not possible to model the dependency of the cells in the gridded representation of the search area on the trail and location of the LP. The search results are illustrated in Figures 2.15d to 2.15g, taking 107 time-steps visiting 103 cells to locate the target.

2.8 Summary

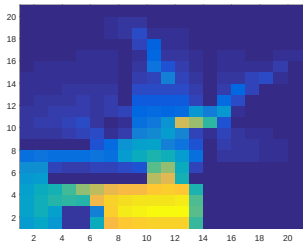
This chapter started by detailing the timeline of WiSAR. Then to automate the search, a probabilistic model of search for a target was detailed, which consists of the initial distribution generation and the update models along with a target detection model and search path planning. Since there are different ways of generating the initial distribution ranging from simple to more complex, we gave examples of some of the simple approaches first and then explained a more sophisticated approach called the diffusion model. We then presented the description of the overall search model using a simple example, comparing it with what actually happens in WiSAR, detailing the limitations of current search models. We argued that current search mod-



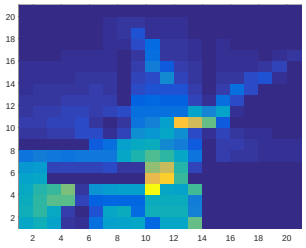
(a) Initial classification of land cover.



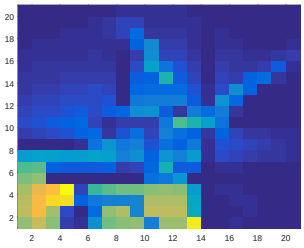
(b) New classification of land cover. Note that the bridge over the river (black over the blue) is missing.



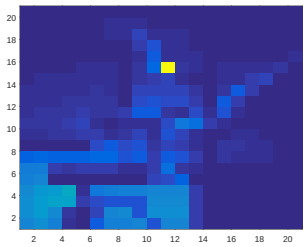
(c) Initial distribution.



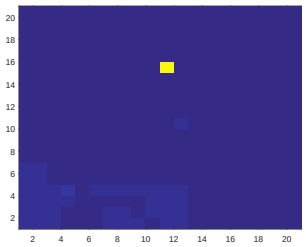
(d) Posterior at $k = 50$.



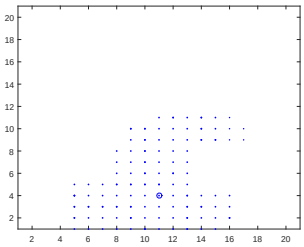
(e) Posterior at $k = 100$.



(f) Posterior at $k = 132$.



(g) Posterior at $k = 135$.



(h) Cells visited.

Figure 2.15: The bridge scenario. The search area consists of two regions (region A representing the area to the left of the river and region C representing the area to the right of the river), which are connected via a bridge b. The blue and black colours represent stream of water and path respectively. The variation in green represents the different densities of vegetation, with light green being sparse vegetation. The rest of the colours represent other types of obstacle. Both PLS and the LP end location are in region A. Figure 2.15b illustrates the current state of the land cover where the bridge no longer exists. Figure 2.15c presents the generated diffusion-based initial distribution. Figures 2.15d to 2.15g present the updated distribution or posterior on the LP location. Each figure represents update of the distribution over the location of the LP after certain amount of time. Figure 2.15h shows the cells visited before the LP was detected.

els have limitations both during the initial distribution generation and then during the search. During the initial distribution generation, key state dependent behaviours exhibited by lost people such as re-orientation strategies and fatigue are ignored. In addition, only local interactions of the LP is considered, which severely limits the capability of the approaches to faithfully represent the LP movements. Then during the search phase, valuable sources of information, such as evidence deposited and land cover changes, are ignored.

To address these, we believe an improved search framework is required.

Chapter 3

A Novel Global Search Framework

3.1 Introduction

In the previous chapter, we argued that existing approaches for Bayesian search often use simple local models of both LP behaviour and observation information. However, this differs from actual search practice where global effects (such as the effect of an obstacle at a distance on the movement of the LP), together with evidence and land cover classification observations are used. We argued that, as a result of this, a new approach to search is needed. We now describe this approach. For ease of presentation we break this into several parts. This chapter provides an overview of the approach and the associated search model. Chapters 4 to 8 describes the implementation of the model in detail.

The global search model is built on the idea that we infer over the entire *trail* of the LP, rather than just the LP's final location. To generate these trails, we need to model the LP's movement and interaction with environment, which can be very complex. Therefore, we first give a general overview of agent-based modelling – an approach capable of capturing complex behaviour [77]; and then define what trails are and describe how the probabilistic search model in Chapter 2 can be rewritten to consider trails rather than current location.

We begin by giving a general overview of Agent-Based Modelling (ABM) and its uses in Section 3.2. We will detail ABM further in the next chapter, presenting review of human movement specific agent models along with the proposed model. We then describe and present the probabilistic model of the trail in Section 3.3. Section 3.4 presents the proposed trail-based search model, capturing the relation between trail, environment and search UAV. Using a graphical model, the trail-based search is probabilistically modelled using a recursive Bayes' formulation. Finally, Section 3.5 summarises the chapter.

3.2 Agent-Based Modelling

ABM is a class of computational models for simulating the actions and interactions of autonomous agents (both individual or collective entities such as organisations or groups) with a view to assessing their effects on the system as a whole [78].

ABM methods have been applied in many areas, ranging from biological, social and physical systems to economics. For example, in biological sciences ABM methods are used to simulate intracellular signalling networks in proteins [79], stem cell behaviour [80], bacterial behaviour and interaction at multiple scales [81]. In social sciences, ABM methods are used to represent a person or group of people and their social interactions [82, 83]. In economics, they have been used to study the development of economy as an evolving systems of autonomous interacting agents [84,85]. They have even been used in anthropology to model the behaviour of ancient civilisations, explaining their growth, decline and demise using archaeological data [86,87].

The theoretical basis of ABM lies mainly in cellular automata, complex system modelling, artificial life, and swarm intelligence [88]. For example, from a complex science perspective, ABMs encompass heterogeneous subsystems or autonomous entities, which often feature nonlinear relationships and multiple interactions [89,90]. Therefore, ABMs are able to represent organised but unpredictable behaviours of natural systems that are considered fundamentally complex. This results in the emergence phenomenon, which is a process whereby larger entities, patterns, and regularities arise through interactions among smaller or simpler entities that themselves do not exhibit those properties [91].

The fundamental feature of an agent is its capability of actively making independent decisions. Other main characteristics include [77,92]:

- *Autonomy:* They are self-directed and independent and make their own decisions.
- *Identifiability:* Each is a discrete individual, with its own set of characteristics and rules that govern its behaviours and decision making capabilities.
- *Situated:* They live in a simulation environment with which they interact.
- *Active:* They are active exerting independent influence in a simulation. They can be:
  - *Goal-directed:* They can have certain goals to achieve.
  - *Perceptive:* They can have an awareness or sense of their surroundings.
  - *Interactive:* They can interact with other agents and / or the environment.

– *Flexible:* They can have the ability to learn from the environment and adapt their behaviour.

In recent years, ABMs have extensively been used to model human movement in a number of environments [83, 90, 93–97]. This has been due to reasons such as:

- *Capability of agents to model natural interaction*: Agents are capable of modelling natural interaction with environment and other agents representing humans movement in the real world [77, 98].

- *Cheaper computational power*: Computational processing power and memory size is advancing and becoming cheaper. As a result, millions of calculations required by some complex ABM methods can be performed in seconds.

- *Comparative ease of acquiring observed data*: Due to advances in machine vision and with improved accuracy and availability of cheap logging equipment like portable GPS systems, it is comparably easy to acquire observed data to tune agent model parameters [99].

When modelling movement of LPs, we need to account for things such as the perceptual state and level of fatigue. As a result, ABM provide ideal framework within which to construct models of how the LP moves.

3.3 Trail-Based Representation of the Lost Person Movement

As discussed in Section 2.7, the most significant problem with grid-based methods of initial distribution generation is that they do not maintain the movement history. This loses much important information. To maintain it, we represent the state by the *entire temporal history* of the LP. This has several advantages. From a modelling perspective, we will show that it makes it possible to use rich and expressive classes of models. From an inference perspective, it becomes possible to use a much wider and more diverse set of information more effectively.

Let $\mathbf{T}$ be the trail of an LP. It is composed of a sequence of vectors,

$$\mathbf{T}_B = \{\mathbf{t}_L, \ldots, \mathbf{t}_S\}, \tag{3.1}$$

where $\mathbf{t}_L$ is the state of the LP at time L and $\mathbf{t}_S$ is final state when the LP became stationary. Because the stopping time S is unknown, the length of the trail is, itself, random. As a shorthand, defining the event $\Lambda = \{P, E, T, V, W\}$, the goal is to compute $p(\mathbf{T}_B|\Lambda)$. From the

Chain Rule (2.2), the LP's movement is given by first order Markov model

$$p(\mathbf{T}_B|\Lambda) = p\left(\mathbf{t}_S|\mathbf{t}_{S-1},\Lambda\right) p\left(\mathbf{t}_{S-1}|\mathbf{t}_{S-2},\Lambda\right) \times \cdots \times p\left(\mathbf{t}_{L+2}|\mathbf{t}_{L+1},\Lambda\right) p\left(\mathbf{t}_L|\Lambda\right). \tag{3.2}$$

This can be written more compactly as

$$p(\mathbf{T}_B|\Lambda) = p\left(\mathbf{t}_L|\Lambda\right) \prod_{\tau=L+1}^{S} p\left(\mathbf{t}_\tau|\mathbf{t}_{\tau-1},\Lambda\right), \tag{3.3}$$

where $p\left(\mathbf{t}_\tau|\mathbf{t}_{\tau-1},\Lambda\right)$ is the state transition density.

Using the generated trail, we can readily define mapping for a grid-based representation of environment. For example, suppose the indicator function is defined

$$I(\mathbf{x},a) = \begin{cases} 1 & \mathbf{x} \in a \\ 0 & \text{otherwise} \end{cases}, \tag{3.4}$$

Using (3.4), we can readily compute the quantities for the grid by integration. For example, the probability that the LP ended up in cell a_i can be computed by

$$p(\mathbf{l}_S = a_i) = \int I\left(\mathbf{t}_S, a_i\right) p(\mathbf{t}_S) \mathrm{d}\mathbf{t}_S. \tag{3.5}$$

Having modelled the trail, we need to decide how to represent the distribution.

3.3.1 Particle Representation

A common representation used to model uncertainty in highly non-linear models is to use particle distributions (reasons detailed in Appendix C). Particle-based methods are often referred to as particle filters [100], bootstrap filters [101], survival of the fittest [102] or the condensation algorithm [103]. The key idea is to represent the density as a set of random samples with associated weights and to compute estimates based on these samples and weights [101]. For example, in case of agent trail

$$p(\mathbf{T}_B) = \{\mathbf{T}_B^{(i)}, w_B^{(t,i)}\}_{i=1}^N \tag{3.6}$$

where the superscript t stands for trail, and $\mathbf{T}_B^{(i)}$ is the i^{th} trail sample with an associated weight $w_B^{(t,i)}$. The weight defines the contribution of the particle to the overall estimate of the variable [104]. To compute the probability densities, estimates of the state can be derived using the state of each particle and magnitude of its weights [104]. As the number of samples N

becomes very large, this particle-based characterisation becomes an equivalent representation to the usual functional description of the posterior probability density function [100].

The weights are recursively updated using observations. As a result, some particles end up having higher weight and others lower. This however can lead to *degeneracy* [100]. This happens when only a few particles have non-negligible weights. To deal with this problem, re-sampling is performed, which replaces those particles with low weights with those with high weights. This however, can lead to another problem. As the re-sampling mechanism eliminates some particles with every iteration, we ultimately end up in a situation where the beginning of every particle trail ultimately becomes the same. This reduction in the number of distinct samples is termed *sample impoverishment* [100]. To address this issue, re-sampling is only performed when it is determined that the particles have become degenerate.

To determine if current collection of samples are degenerate, we use the effective sample size N_{eff},

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} \left(w_k^{(t,i)} \right)^2}. \tag{3.7}$$

The term N_{eff} provides a measure of how uniformly distributed the weights of the particles are. If all weights are the same $w_k^{(t,i)} = \frac{1}{N}$

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} \left(w_k^{(t,i)} \right)^2} = \frac{1}{N \frac{1}{N^2}} = N, \tag{3.8}$$

but if all particle weights are zeros except one

$$N_{eff} = \frac{1}{1} = 1. \tag{3.9}$$

The smaller N_{eff} is, the greater the imbalance in the particle weights. Therefore, re-sampling is carried out when $N_{eff} < \eta N$, where η is some factor i.e. 0.5.

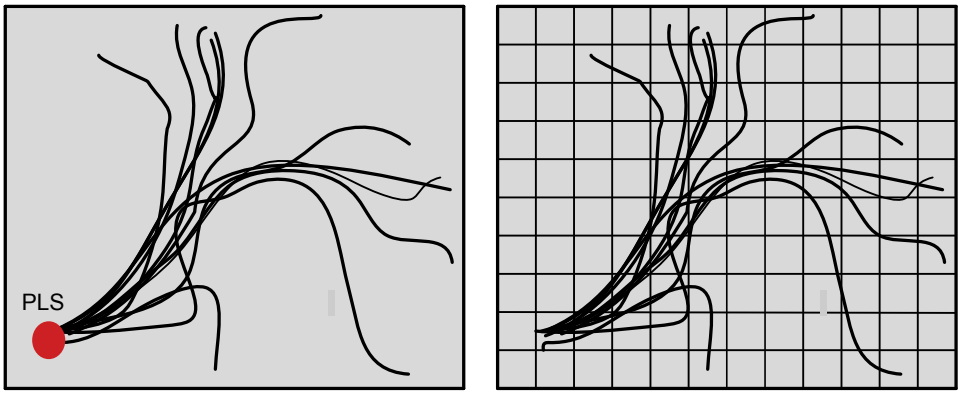
3.3.2 Particle Representation of Trail

Each particle represents a candidate hypothesis trail,

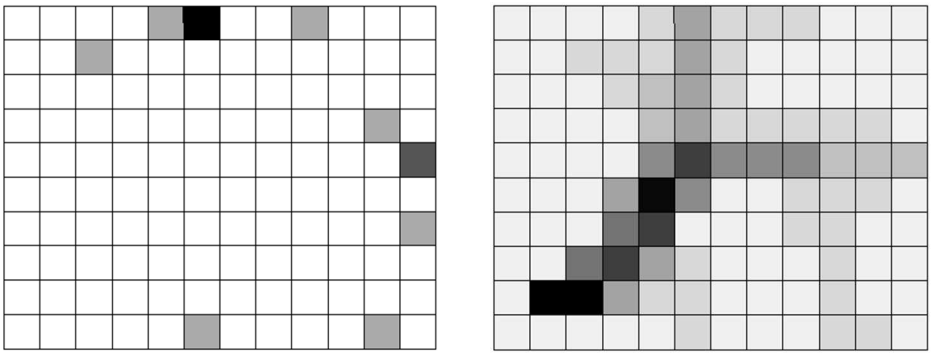
$$p\left(\mathbf{T}_B | \Lambda\right) = \sum_{i=1}^{N} w_B^{(t,i)} \delta \left(\mathbf{T}_B - \mathbf{T}_B^{(i)} \right), \tag{3.10}$$

where $\delta(\cdot)$ is a suitably defined delta function.

With this representation, the method for computing probability of the LP being in a_i is



(a) Agent trails **(b)** Discretised search area



(c) Distribution over end location $p(\mathbf{l}_B = a_i)$ **(d)** Distribution over trajectory$p(a_i \in \mathbf{T}_B)$

Figure 3.1: Particle-based distribution over the LP end location and trail.

given by

$$p(\mathbf{l}_B = a_i) = \sum_{j=1}^{N} w_B^{(t,j)} I\left(\mathbf{t}_B^{(j)}, a_i\right), \tag{3.11}$$

where I was defined in (3.4).

Similarly let $T(\mathbf{T}_B^{(j)}, a_i)$ be an indicator function which takes the value 1 if cell a_i lies on trail $\mathbf{T}_B^{(j)}$ and 0 otherwise. The probability that a cell a_i lies on the path taken by the LP is

$$p(a_i \in \mathbf{T}_B) = \sum_{j=1}^{N} w_B^{(t,j)} T\left(\mathbf{T}_B^{(j)}, a_i\right). \tag{3.12}$$

We will show later in this thesis that this particle representation of the LP trajectory can be used in two ways: to help direct the search process (by identifying potential trails the LP took), and enabling the use of environment observations to indirectly update the location of the LP.

The process of computing (3.11) and (3.12) is illustrated in Figure 3.1. Figure 3.1a shows a series of generated trails which are overlaid in the grid Figure 3.1b. Figures 3.1c and 3.1d show (3.11) and (3.12) respectively. Figure 3.1c shows that there are several locations where the LP is likely to end up. Figures 3.1d shows the probability that cells lie on a path. As can be

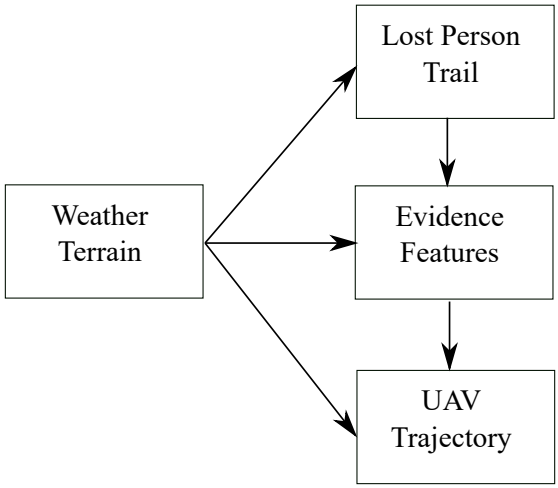


Figure 3.2: The top-level representation of interplay between the environment (weather, land cover and terrain), the LP trail, the trajectory of the UAV, and the evidence features.

seen, the cells close to the PLS have a strong probability of lying on the path, but this rapidly diffuses the further a cell is from the PLS.

3.4 Trail-Based Search Model

The search area conditions play a pivotal role in the search for a LP. For example, Section 2.7.2 described how the environment information can be used to compute an initial distribution on LP's movement. This distribution can be used to prioritise the UAV search path.

At a top-level the interplay between the search area $\mathcal{A}$, trail $\mathbf{T}$ and the UAV is illustrated in Figure 3.2. The weather conditions and the terrain effect the LP's movement and ability to lay down evidence. The terrain and weather also effect the UAV's ability to detect and classify the features.

To model these effects, we need to first describe the composition of the search and then probabilistically model it.

3.4.1 Composition of the Search Model

To generate the initial distribution, we consider the complete set of information acquired during the information gathering phase of WiSAR (Section 2.2). To model the search area, we use a Digital Surface Model (DSM) as well as the environmental data sets listed in Section 2.5.2. The DSM, like the Digital Terrain Model (DTM), attempts to model the shape of the environment. However, the DTM attempts to model the shape of the underlying land, whereas the DSM attempts to model the effects of permanent structures as well. Figure 3.3 illustrates the difference between the two - the DTM models the land cover under the trees, whereas the DSM includes the trees as well.

(a) DSM **(b)** Actual image **(c)** DTM

Figure 3.3: Difference between the two elevation models (DTM and DSM). DSM considers the added height of permanent features like trees and buildings.

In the search phase, the observation model detailed in previous chapter is extended to include two new types of observations information:

1. A set $\mathbf{m} = \{\mathbf{e}, \mathbf{n}\}$ of *point-like features* where:

   - $\mathbf{e}$ *is the evidence features,*. These include items such as discarded clothes and food wrapper, some of which are illustrated in Figure 1.5. Multiple evidence features can be left along the trail.
   - $\mathbf{n}$ *is the non-evidence features,* These are other features that can be detected by the UAV but are not directly related to the movement of the LP. Examples could be rocks or trees. Non-evidence features are a source of false positive evidence features. However, they also provide landmarks which could be used to aid in localising the UAV.

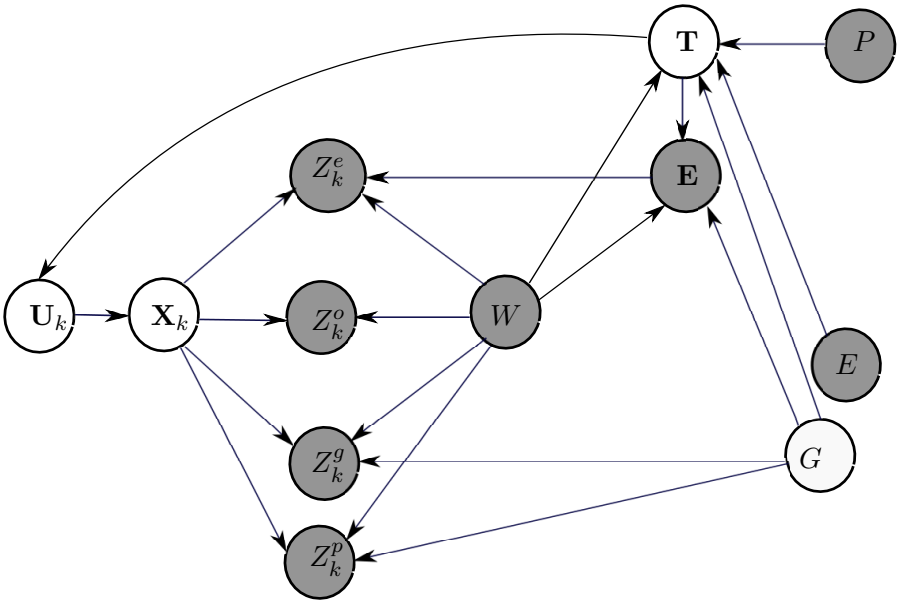
2. *Observations of land cover* $G = \{V, T\}$. This includes reclassification of the land cover. An example of this was illustrated in the synthetic scenario in Figure 2.15.

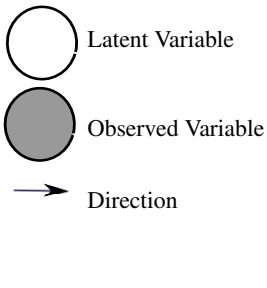
As a result, defining Z_k to be the history of observations up to time-step k i.e. $Z_k = \{z_1, \cdots, z_k\} = \{Z_{k-1}, z_k\}$, $Z_k = Z_k^p \bigcap Z_k^e \bigcap Z_k^g \bigcap Z_k^o$. The superscripts p, e, g, o stand for the LP, evidence, land cover type and others. The latter includes observation of things like non-evidence features, GPS and IMU. Further explanation of these information types and how they are modelled will be given in later chapters.

3.4.2 Graphical Model of the Search Model

The graphical model of trail-based search is shown in Figure 3.4. In this model, $\mathbf{T}$ is a function of the profile P, weather W, terrain elevation data set E and land cover type G. While traversing, it is assumed that the LP deposits / leaves evidence subject to G and W.

The movement of the UAV with state $\mathbf{x}_k \in \mathbf{X}_k : \mathbf{X}_k = \{\mathbf{x}_1, \cdots, \mathbf{x}_k\} = \{\mathbf{X}_{k-1}, \mathbf{x}_k\}$ is controlled by $\mathbf{u}_k \in \mathbf{U}_k : \mathbf{U}_k = \{\mathbf{u}_1, \cdots, \mathbf{u}_k\} = \{\mathbf{U}_{k-1}, \mathbf{u}_k\}$. Since the UAV is undertaking a





P - Profile of the lost person
$\mathbf{T}$ - The lost person trail
$\mathbf{X}_k$ - History of UAV states
$\mathbf{U}_k$ - History of UAV control inputs
$\mathbf{E}$ - Set of evidence features

W -The weather condition
E- Elevation data-set of the search area
G - Ground Classificaiton
Z_k^p - History of Lost Person observations
Z_k^e - History of evidence features observations
Z_k^g - History of ground observations
Z_k^o - History of other observations

Figure 3.4: Graphical model of the proposed search framework showing the overall interactions between the observed (filled symbols) and latent variables (clear symbols). The LP trail $\mathbf{T}$ is affected by factors including: the profile P, the elevation and classification of the search area E and G respectively. Evidence $\mathbf{E}$ and land cover classification G are producers of observation. Depositing $\mathbf{E}$ is a function the LP's trail $\mathbf{T}$ and the LP's ability to deposit evidence on the ground is effected by both G and W. UAV observations Z_k consist of four types: evidence, the LP, land cover classification and other (salient features, GPS, IMU) given as Z_k^e,Z_k^p, Z_k^g and Z_k^o respectively. The UAV observations are effected by the platform altitude given by $\mathbf{X}_k$, the weather W.

search task, the control inputs will be a function of the evolving distribution over $\mathbf{T}$. The UAV captures the search area and takes measurements which includes: Z^e, Z^g, Z^p and Z^o. While Z^p, Z^e and Z^g are used to infer $\mathbf{T}$, Z^o is typically used to localise the search platform and construct a map of the environment.

3.4.3 Probabilistic Model of Search

Similar to the search model presented in Section 2.3, the update is performed recursively using Bayes' rule, until either the LP is found or the search is terminated.

Given the observation history Z_k and our initial knowledge of the Λ, the goal is compute the trail taken by the LP.

$$p\left(\mathbf{T}|\Lambda, Z_k\right), \tag{3.13}$$

This requires $p(\mathbf{T}_B|\Lambda)$ as the initial estimate of the LP trail or prior distribution to be available at the beginning of search operation. However, there are three factors one has to consider when modelling this. First, new information about the search area is collected from the UAV whose pose history is $\mathbf{X}_k$. Second, as discussed earlier, we need to consider new observations of land cover type. Third, we need to build a map of evidence and non-evidence features detected to be used in the trail inference and localisation. Therefore, the full state of the problem in (3.13) requiring inference can be extended to

$$p(\mathbf{T}, G, \mathbf{m}, \mathbf{X}_k|\Lambda, Z_k, \mathbf{x}_B), \tag{3.14}$$

where $\mathbf{x}_B$ is the state of the UAV at the time of aerial search initiation. Since we are only interested in the current states of the search platform, we can re-write (3.14) as

$$p(\mathbf{T}, G, \mathbf{m}, \mathbf{x}_k|\Lambda, Z_k, \mathbf{x}_B). \tag{3.15}$$

This can be computed recursively using *predict* and *update* steps to model the uncertainty in search platform position. Assuming that the UAV process model is Markov, the predict step is computed by

$$\begin{aligned}
p(\mathbf{T}, G, \mathbf{m}, \mathbf{x}_k|\Lambda, Z_{k-1}, \mathbf{x}_B) &= \int p(\mathbf{T}, G, \mathbf{m}, \mathbf{x}_k, \mathbf{x}_{k-1}|\Lambda, Z_{k-1}, \mathbf{x}_B)\, d\mathbf{x}_{k-1} \\
&= \int p(\mathbf{x}_k|\mathbf{T}, G, \mathbf{m}, \mathbf{x}_{k-1}, \Lambda, Z_{k-1}, \mathbf{x}_B) \\
&\times p(\mathbf{T}, G, \mathbf{m}, \mathbf{x}_{k-1}|\Lambda, Z_{k-1}, \mathbf{x}_B)\, d\mathbf{x}_{k-1} \\
&= \int\int p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)\, p(\mathbf{u}_k|\mathbf{T}, \mathbf{x}_{k-1}) \\
&\times p(\mathbf{T}, G, \mathbf{m}, \mathbf{x}_{k-1}|\Lambda, Z_{k-1}, \mathbf{x}_B)\, d\mathbf{u}_k d\mathbf{x}_{k-1} \\
&= \int\int p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)\, p(\mathbf{u}_k|\mathbf{T}, \mathbf{x}_{k-1}) \\
&\times p(\mathbf{T}|G, \mathbf{x}_{k-1}, \Lambda)\, p(G, \mathbf{m}, \mathbf{x}_{k-1}|\Lambda, Z_{k-1}, \mathbf{x}_B)\, d\mathbf{u}_k d\mathbf{x}_{k-1}.
\end{aligned} \tag{3.16}$$

The first term on the right hand side predicts the platform state at time-step k given that we know where it was in the previous time-step $k-1$ and the latest control input computed by the second term using the estimate of the LP trail given observations up to time-step $k-1$. The third term $p(\mathbf{T}|G, \mathbf{x}_{k-1}, \Lambda)$ models the affects of latest land cover classification on the trails. The fourth term is simply the estimate of the land cover type and platform from previous time-step

given observations up until time-step $k-1$.

The update is then computed using Bayes' rule,

$$p\left(\mathbf{T}, G, \mathbf{m}, \mathbf{x}_k | \Lambda, Z_k, \mathbf{x}_B\right) = \eta \cdot p\left(z_k | \mathbf{T}, G, \mathbf{m}, \mathbf{x}_k, \Lambda\right) p\left(\mathbf{T}, G, \mathbf{m}, \mathbf{x}_k | \Lambda, Z_{k-1}, \mathbf{x}_B\right), \quad (3.17)$$

where $p\left(z_k | \mathbf{T}, G, \mathbf{m}, \mathbf{x}_k, \Lambda\right)$ is the observation model, describing the probability of making an observation z_k when the true state of the world is $\{\mathbf{T}, G, \mathbf{x}_k, \mathbf{m}, \Lambda\}$, and η is the normalising constant.

Since $z_k = z_k^p \bigcap z_k^e \bigcap z_k^g \bigcap z_k^o$ are conditionally independent given the platform state, the observation likelihood model in (3.17) can be decomposed into

$$\begin{aligned} p\left(z_k | \mathbf{T}, G, \mathbf{m}, \mathbf{x}_k, \Lambda\right) =& p\left(z_k^p | \mathbf{T}, G, \mathbf{m}, \mathbf{x}_k, \Lambda\right) p\left(z_k^e | \mathbf{T}, G, \mathbf{m}, \mathbf{x}_k, \Lambda\right) p\left(z_k^g | \mathbf{T}, G, \mathbf{m}, \mathbf{x}_k, \Lambda\right) \\ & \times p\left(z_k^o | \mathbf{T}, G, \mathbf{m}, \mathbf{x}_k, \Lambda\right) \\ =& p\left(z_k^p | \mathbf{T}, G, \mathbf{x}_k, W\right) p\left(z_k^e | \mathbf{T}, G, \mathbf{x}_k, W, P, \mathbf{E}\right) p\left(z_k^g | G, \mathbf{x}_k\right) p\left(z_k^o | \mathbf{x}_k, \mathbf{N}\right), \end{aligned} \quad (3.18)$$

where the terms on the right are the LP, evidence, land cover type and other observations likelihood models respectively.

3.4.4 The Proposed Search Framework Process

Our proposed search model consists of three key phases: Information gathering, initial distribution generation, and search. The dependency of the phases and flow of the search process is illustrated in Figure 3.5.

When a person is reported missing, information is gathered in the same way as before. Next, using the information gathered, an initial distribution over the trail $\mathbf{T}$ is generated. This is achieved using a LP movement-based agent model.

The search phase is initiated by deploying a UAV to search for the LP. This phase consists of several interrelated tasks, which are: UAV path planning, moving UAV to a new way-point, detecting and classifying UAV observations, localising the UAV and features detected and finally updating the distribution over the LP trail.

The initial distribution over the LP trail is recursively updated with new observations of the search area. This evolving probability distribution over the LP's trail is assessed after every update step for detection of the LP. We declare that the LP has been found if $\max_{a_i \in \mathcal{A}} p(\mathbf{t}_S = a_i) \geq \gamma$ and the search process ends. If not, the search continues until this condition is satisfied or the search is terminated.

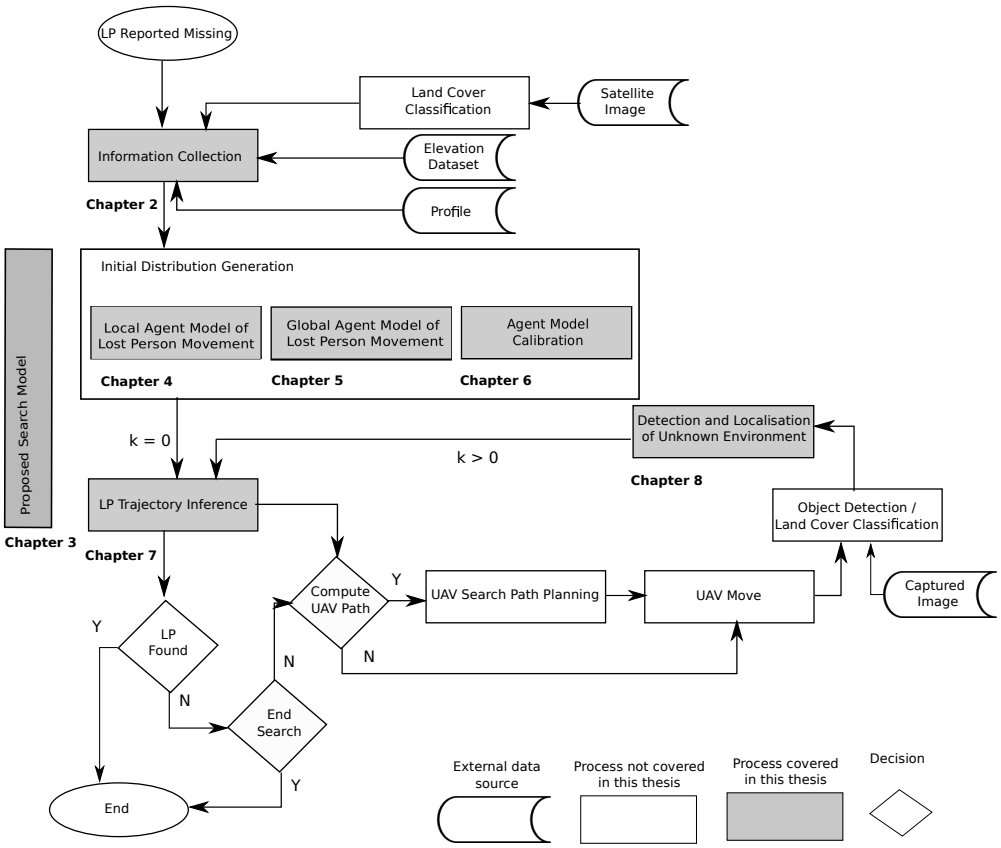


Figure 3.5: A more detailed version of automated agent trail-based search process flow diagram.

3.5 Summary

In this chapter we proposed a new search framework. The novelty of the proposed search model is that it considers the LP trail (history of movements) rather than just the current location.

We began by review of ABM method mainly used to model complex behaviours. We then described and probabilistically modelled the LP trail and discussed its representation. We argued that since LP motion can be highly non-linear, particles representation is best suited.

We then presented the trail-based search model, describing the composition of the search model, and the probabilistic representation of it. The latter, using a high level graphical model, showing how different model variables interacted and how they all contributed in inference of the LP's trail. Having presented the proposed search framework, we then detailed the process of search using the framework with the help of a flow diagram.

The next five chapters derive components of this model in greater detail. In Chapters 4 and 5 we model the LP movement and decision making ($p\left(\mathbf{T}_B|\Lambda\right)$ in (3.13)), first at a local level, and then extending it to consider features at a global level. At local level, the agent model will consider features in the neighbourhood similar to the diffusion model. At global level, the agent will consider features located both local and at a distance (non-local features).

Figure 3.6: A deployed UAV flying over rugged terrain.

To ensure the developed model is able to produce faithful representation of the LP movements, Chapter 6 details the process of tuning its parameters using observed data. Chapter 7 then describes the second problem: inference of the LP trail based on observations of the land cover (Equations (3.16) and (3.17)) modelling the first three terms on the right hand side of (3.18) assuming known UAV pose $\mathbf{x}_k$. Finally, having detailed the initial distribution generation and search phases, Chapter 8 models the fourth term and describes the localisation method used to localise the detected evidence features and presents results of running the proposed search model as a whole.

Chapter 4

Local Agent Model of Lost Person Movement

4.1 Introduction

In this chapter, we develop an agent model which captures local influences on the LP's movement[1]. It serves as an introduction to both ABMs and develops the basic infrastructure which is extended in Chapter 5 to include global effects.

We begin by reviewing hiker movement-based agent models in Section 4.2. The design of the proposed agent model is described in Section 4.3. Section 4.4 then details the implementation of the agent model with encoded behaviour similar to the diffusion model. To show the agent behaves as expected, its encoded behaviour is evaluated in Section 4.5. To evaluate the trails generated using the local agent model, we need to compare them with movement data of lost people in reality. However due to issues with data availability, we perform a data collection experiment detailed in Section 4.6. In this experiment we collect GPS logs of participants movements in a wilderness like environment. Then in Section 4.7, we evaluate the local agent model, comparing the trails generated with movement pattern of experiment participants. A summary of the chapter is provided in Section 4.8.

4.2 Agent-Based Human Movement Model

Most ABM methods of human movement have been developed to simulate and study the behaviour of people in urban environments. This includes the movement of individuals in galleries [105], the movements of crowds in normal [106, 107] or emergency [108] situations and individual pedestrian movement in large areas such as sport centres [94]. Most of these models are used to study the social interactions between people, and are not relevant for Wilderness Search and Rescue.

The first documented implementation of human movement-based agent model in wilder-

[1] In a discretised environment, local influence means influence of features in the neighbouring cells. We will cover global influences that considers non-local features in the next chapter.

ness is in [109]. This model was based on idea that a visitor's enjoyment of wilderness experiences was inversely related to the number of encounters with other visitors. Therefore, this model modelled and measured the times re-creationists interacted with each other, the types of encounter, location of encounters and methods of travel. The output from the model was used to control trail access, such as limiting the number of visitors entering a wilderness area over a given time period, or staggering their en-tries.

The simulation uses a very simple network representation of the recreation area trails and camp sites and requires the modeller to input summary data about groups of re-creationists [110]. This included information about their party size, start location, end location, start time, and travel speed. In order to represent the attractiveness of particular routes, modellers are required to input the probability that re-creationists would chose a given path. Based on these parameters, the model generates random groups of re-creationists The model simulates a set period of time (typically a day), and inserts hikers into the network of trails at an appropriate time. The re-creationists move at a constant rate along segments, and chose between paths using a weighted random choice whenever they reach a junction.

More recently, in [83, 93], Gloor et al. explored the feasibility of using ABM to evaluate future scenarios in a tourist landscape in the Swiss Alps. The project uses simulated people (agents) with perception capabilities, traversing and interacting with the environment using a pre-planned activity plan generated at the start of simulation for each agent. The aim is to test their reactions against the simulated scenarios to evaluate the long term implications of a future planning decision on tourism.

Although these models account for various factors including occlusion and lack of initial knowledge about the environment, none of them can account for a person being lost. In particular, they assume that the agent knows where it is, and knows where its destination is. However, in WiSAR, people become lost, unsure of where they are and where to go. The only model we are aware of that attempts to capture this behaviour is Goodrich's diffusion model [61], which was explained in Section 2.4. However, this model fails to account or consider many important issues and behaviours discussed in Section 2.7.

Furthermore, the common assumption in most of the hiker movement-based agent models is that people always follow natural or predefined paths and that they only perform route selection when faced with a junction. This is not the case in WiSAR. LPs actually traverse the environment by switching between a number of re-orientation strategies [3, 74]. For example, an LP that normally follows a path of least resistance, may head towards a particular direction when a place of interest such as a house is observed.

This is reflected by [98], which states that an issue common to all modelling techniques is that a model has to serve a purpose; a general-purpose model cannot work. The model has to be built at the right level of description, with just the right amount of detail to serve its purpose. As a result, we need to design and model a new agent model that is capable of representing the LP movements.

4.2.1 Agent-Based Human Modelling Approaches

ABM is synonymous to microscopic modelling [98, 106, 111, 112]. Microscopic models can generally be categorised into three groups [113, 114], which are partially overlapping [112]:

- *Social Force Models (SFMs):* The fundamental concept behind this type of model is that humans feel social force either repulsive or attractive when in close proximity to other humans or to environment objects such as obstacles that impact their reaction and decision making. SFM captures this using models such as those in [83, 115]. With these models, the speed of the agent is influenced by forces in the environment such as attractive force of places of interest and repulsive force of obstacles, boundaries and other people. This form of human movement modelling is mainly used in simulated panic situations such as emergency evacuation [116, 117] or movement in small built areas where the behaviour modelled considers moving from point A to point B (known exits) avoiding obstacles [118] and adjusting speed and direction to model queuing, lane formation [119]. Its use in large outdoor environments has mainly been for the purpose of path following behaviour and obstacle avoidance [83].

- *Cellular Models:* Cellular models are based on discrete space and time. Each spatial unit is called a cell, and can either be occupied by one pedestrian or obstacle, or be empty. This is a variant of the traditional cellular automata models (eg. Conway's Game of Life [120]) where cells have a state which changes depending on the state of the surrounding cells and there is no explicit movement involved. In this model, the agent movement is modelled with a matrix of preferences with each cell containing move probability values. The model evolves in discrete time steps. The variables at each cell are updated simultaneously based on the values of the variables in their neighbourhood at the previous time step and according to a set of rules [121]. The preference matrix is used to capture interactions between agents, preferences and obstacle avoidance, and is computed using methods like floor field [122, 123], vector fields [97] and potential fields [124]. While very effective for small areas with known exit points or goals such as those in [96, 113], cellular models are dependent on the discretisation of the space and can be computationally very expensive,

requiring multiple force vector fields to represent differing goals [124].

- *Rule-based Models / Agent-based Models:* This includes autonomous agents that can move either continuous or discrete both in space and time and are governed by behavioural rules (more complex than cell-based models [112]). Rule-based models often have a large set of behavioural rules, each dedicated to a specific situation [114]. A person can be modelled with own self-contained behaviours. Each individual can be given individual goals, attributes, perception capability and reasoning when faced to make decisions. In this model, agents move following a two step process: First, the agent mostly determines the situation it is in, and then executes the rule connected to that situation [112].

  To model uncertainties, agent models allow for addition of randomness to specific part of the model [98]. For example, to model uncertainties in the person's speed and orientation and behaviours like preference of moving uphill/downhill.

  The most well known rule-based model to simulate life-like complex behaviour is Reynolds' local rules boids model [125]. In this model, the aggregate motion of the simulated flock is created by a distributed behavioural model. Each agent is implemented as an independent actor that navigates according to its local perception of the dynamic environment, the laws of simulated physics that rule its motion, and a set of behaviours programmed into it. The aggregate motion of the simulated flock is the result of the dense interaction of the relatively simple behaviours of the individual simulated boids. The basic model to simulate generic flocking behaviour consists of three simple rules which describe how an individual boid manoeuvres based on the positions and velocities of its nearby flock mates:

  - Separation: Steer to avoid crowding local flock mates.
  - Alignment: Steer towards the average heading of local flock mates.
  - Cohesion: Steer towards the average position of local flock mates.

  Each boid has access to the whole environment description, but flocking only requires reaction within a specific neighbourhood, which is given by a distance (from the centre of each boid) and an angle (from each boid's direction of flight). This neighbourhood can be considered as limited perception. Each boid avoids not only collision against other boids but also with obstacles in the environment.

  Other examples include [73, 126, 127], where in [126] the agents (representing groups of

river rafters) make decisions about camping sites and stop locations based on the attractiveness of the site and the number of people currently visiting that site.

Since we are interested in modelling an LP – a distinct entity who is autonomous and has abilities to perceive, interact with environment that can be very large, make decisions based on local and global features observed with some associated uncertainty and moves in continuous space, we will use the rule-based / agent-based approach.

We propose to use a state dependent parametric rule-based agent model with capabilities such as state dependent perception, behaviour and decision making. The agent model parameters encompass both LP's mechanical attributes such as position, height, walking speed and bearing angle; and preferences like moving uphill / downhill. Similarly, the rule set contains the LP's state dependent behaviour and re-orientation strategies.

To model the uncertainty in our knowledge of the LP behaviour, similar to the diffusion model detailed in Section 2.5, we specify agent model parameters using probability distributions. This way, initialising a population of N agent particles representing the LP, with sampled behaviour parameters, each agent will have its own unique attributes and rule set, and independent capabilities such as decision making and response to events. Aggregate of the generated agent trajectories (particles) will help determine the distribution over the LP trail as detailed in Section 3.3.2.

4.3 Agent Model Design

The design of the agent model is illustrated in Figure 4.1. As can be seen, similar to the diffusion model, our proposed design has 4 layers: *environment*, *perception*, *movement* and *strategies*. The key difference compared with the diffusion model is the way we handle the relation between the layers. In contrast to the diffusion model where movement is only affected by perception of the neighbourhood cells (the *local environment*), in this design, the agent's overall behaviour is controlled by a set of *strategies*, which control the agent's *movement*. These strategies include moving towards a defined feature or point, exploring the environment to acquire further information, moving randomly or simply stopping. These are driven by the agent's *perception* of the *environment* both close neighbourhood and far (local and global).

Another difference is that it models the fact that movement can be directly influenced by the environment. For example, as explained in Section 2.7, slope has direct impact on an LP's movement or speed.

We now describe the model by considering each layer in turn.

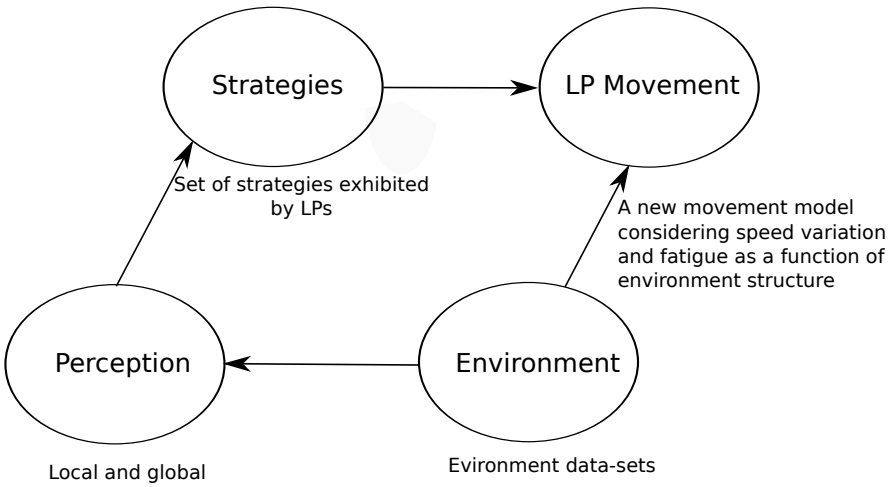


Figure 4.1: The dependency of the layers in the four layer agent design structure.

4.3.1 Environment

As mentioned earlier, the environment plays a crucial role in modelling the human movement. Because we consider the effects of search area elevation on the agent's movement, we use a 2.5D representation of the environment. Agent movements within the continuous environment is time discretised. Each discrete time-step represents a footstep.

4.3.2 Perception

Perception is the processes by which the agent monitors the environment and interprets its meaning. Human movement and actions are strongly influenced by the continued perception of the environment and the spatial knowledge of features within it. From a pure visual perspective, people move in the direction that provides them with the potential for further movement. Such interaction between human and environment is called 'natural vision' by Gibson [128]. We will refer to the area visible to an agent as *view area*.

4.3.3 Strategies

Lost people rarely move in a completely random fashion. Rather, LPs tend to rely on simple rule of thumb when attempting to regain orientation. They are typically driven by strategies which they believe will guide them to a known place or to safety [74].

Koester identifies Ten different strategies that lost people can execute [3], six of which are relevant for our problem domain[2]. These strategies can be organised into three groups:

1. **Stationary:** The agent has no goal and does not move. This consists of just the *Stay Put* strategy.

[2]Although rout sampling strategy was initially considered, because it was rarely executed and did not have significant affect on the result, it is not detailed here.

- *Stay Put*: After realising he has become lost, the LP stays in one place with hopes of being found by others. This strategy is often thought of as too passive and is rarely used [3] .

2. **Clear intention to move to a particular point:** The agent has identified a particular point in the environment and is heading to it. This includes:

   - *Direction Travelling*: While performing this strategy, the LP travels towards a place of interest, which can require going off path. This behaviour is only restricted by non navigable areas in the environment such as very dense vegetation or water, which the LP would try to circumvent.

   - *Backtracking*: While trying to re-orient, lost people sometimes consider backtracking their steps along their original route. However most lost people are too reluctant to backtrack without any reason because of the belief that they are very near to a known location or safety and might turn away from it [3]. Our agent model is designed to perform this step under two conditions: First, when the agent is faced with obstacles and cannot move forward or to the sides. Second, it is performed as part of route sampling.

3. **No clear intention to move to a particular point:** The agent does not have a specific point to go to. Strategies included in this group are:

   - *View Enhancing*: The agent moves towards higher ground. This re-orientation strategy is normally used when the terrain is made up of low and high grounds.

   - *Random Travelling*: This strategy is used when the LP is totally confused or has no other option. The LP traverses the environment following the path of least resistance, with no apparent goal other than to find something or some place that looks familiar.

   - *Route Travelling*: This strategy consist of moving along a path. The LP tend to follow paths even if they are new and not known to them. They follow it with the hope that it would eventually lead them to somewhere familiar or to safety. Individuals often revert back to random travel if this is unsuccessful (after a certain amount of time) or when the route ends.

While strategies with no clear intention to move to a particular point are encoded in the transition matrices (2.15), (2.16) and (2.17). Strategies with clear intention to move to a particular

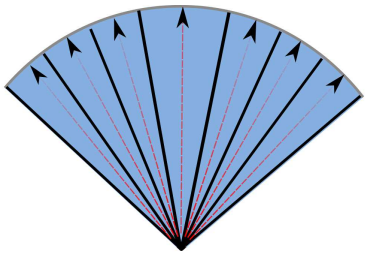


Figure 4.2: The view area discretised into bins. The red arrows inside each bin represent agent's movement rays.

point and stay put strategy require the agent to make a decision, therefore are encoded using decision variables.

4.3.4 Movement

Each agent trail consists of a finite collection of continuous-valued vectors which show a set of positions through which the LP has passed (3.1). Each agent starts from PLS and traverses the search area, interacting with environment steering away from obstacles and towards traversable areas.

To model the steering behaviour, inspired by [72, 106] we decompose the agent's view area into bins illustrated in Figure 4.2. Suppose θ_τ is the direction within view area that the agent could take in the next time-step. This can take on one of a finite number of values, each represented as a central ray in each bin within the view area. Defining b to be the number of bins in agent's view area, the next position is then selected along the i^{th} direction with value $\theta_\tau^{(i)}$ that maximises agent's state transition density,

$$\mathbf{t}_\tau = \arg\max_{\theta_\tau} p\left(\mathbf{t}_\tau | \mathbf{t}_{\tau-1}, \theta_\tau^{(i)}, \Lambda\right). \tag{4.1}$$

To compute $p\left(\mathbf{t}_\tau | \mathbf{t}_{\tau-1}, \theta_\tau^{(i)}, \Lambda\right)$, we use a *propose—accept* approach similar to the approach detailed in Section 2.4. The next proposed position of the agent, $\tilde{\mathbf{t}}_\tau$, is proposed using a kinematics model. The event that this move is accepted is A_τ. The probability of its acceptance is governed by the move being consistent with both the agent's strategy and the constraints placed by the environment. Therefore similar to (2.12), $\mathbf{t}_\tau = \{\tilde{\mathbf{t}}_\tau, A_\tau\}$, specifically,

$$\begin{aligned} p\left(\mathbf{t}_\tau | \mathbf{t}_{\tau-1}, \theta_\tau^{(i)}, \Lambda\right) &= p\left(\tilde{\mathbf{t}}_\tau, A_\tau | \mathbf{t}_{\tau-1}, \theta_\tau^{(i)}, \Lambda\right) \\ &= p\left(A_\tau | \tilde{\mathbf{t}}_\tau, \mathbf{t}_{\tau-1}, \Lambda\right) p\left(\tilde{\mathbf{t}}_\tau | \mathbf{t}_{\tau-1}, \theta_\tau^{(i)}, \Lambda\right). \end{aligned} \tag{4.2}$$

Having proposed the agent model of the LP movement, we need to implement it. Implementation of agent model can be very complex. We start by presenting a *local agent model* with

only local environment interactions. Then, in the next Chapter we will build on this model and introduce new features to it allowing for global interactions. We call this *global agent model*.

4.4 Local Agent Model

With the local agent model, our aim is to model the LP's movement and local interaction with environment. We achieve the later by translating the behaviours modelled in the diffusion-based model presented in section 2.5. The following subsections, describes the detail of the local agent model design layers.

4.4.1 Agent Perception

For the local agent model, agent's perception consists of its vision. The vision is based on how far and wide into the environment it can detect. Inspired by [72, 73, 129, 130], we model the structure of an agent's view area using a circle sector illustrated in Figure 4.2. The view area is parameterised by two variables:

1. **Field Of View (FOV).** FOV is the sweep angle ϑ_τ over which features in the environment are observable to an agent from a location and bearing.

2. **Vision Visual Critical Length (VCL).** This models the maximum distance at which a feature can be seen and considered in agent movement. This is defined by the length l_τ.

As mentioned before, to compute agent's movements, we assume that the agent's FOV (ϑ_τ) is discretised into b non overlapping cones, each one with central ray $\theta_\tau^{(i)}$, along which the next pose is proposed and assessed with respect the features in the environment.

For this local agent model, the agent interacts with environment at a local level. Therefore, l_τ is limited to neighbourhood cells illustrated in Figure 4.3. The agent's view area is centred on the its current position and oriented in its current direction.

4.4.2 Agent Strategies

To replicate the behaviour in diffusion-based model, the local agent model only performs strategies encoded within the transition matrices (2.15), (2.16) and (2.17). These strategies include: view enhancing, random travelling and route travelling. Using these strategies, behaviours such as local obstacle avoidance and following the path of least resistant will be modelled.

4.4.3 Agent Movement

To model the agent movement described in Section 4.3.4, we first introduce the kinematics model of agent movement and then detail the acceptance computation.

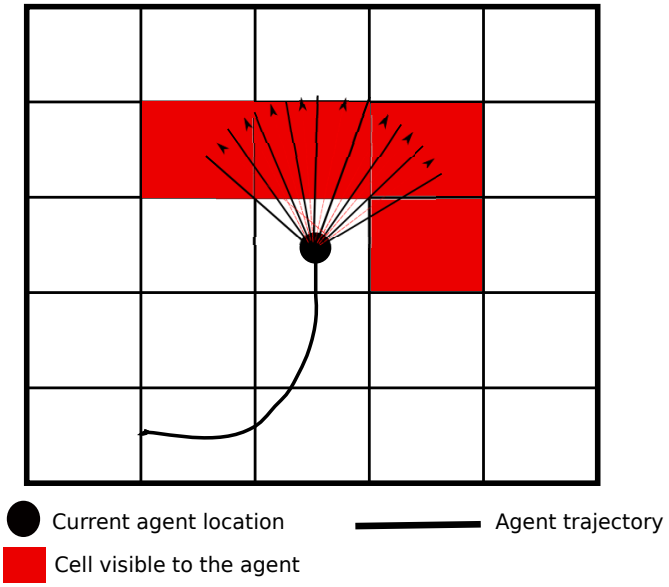


Figure 4.3: Agent's local view area, which depended on agent's current heading.

4.4.3.1 Agent State and kinematics Model

Let $\mathbf{t}_\tau$ be the state of the agent at time τ with the structure[3]

$$\mathbf{t}_\tau = \begin{bmatrix} x & y & \lambda & s & d \end{bmatrix}_\tau^T, \tag{4.3}$$

where $[x, y, \lambda]_\tau^T$ is the kinematic state of the agent, which consists of the agent's Cartesian position (x_τ and y_τ) and it's heading λ_τ, s_τ is the speed of the agent and d_τ is the total distance travelled up until time-step τ. At the start of simulation, the agent's state is initialised to

$$\mathbf{t}_L = \begin{bmatrix} x & y & \lambda & s & 0 \end{bmatrix}_L^T, \tag{4.4}$$

where x_L and y_L are the 2D Cartesian coordinate of the PLS. The agent bearing and speed are sampled from distributions over reported bearing and speed of the LP $\lambda_L \sim G(\mu_{\lambda_{PLS}}, \sigma^2_{\lambda_{PLS}})$ and $s_L \sim G\left(\mu_{s_{PLS}}, \sigma^2_{s_{PLS}}\right)$ respectively. The total distance travelled is 0m.

By default LPs tend to walk in a straight line with constant motion [3]. Also, because most person-based decisions for navigation are carried out in a person-centric frame, the motion model uses piece-wise constant speed and heading. Therefore to propose $\tilde{\mathbf{t}}_\tau^{(i)}$ along movement ray angle $\theta_\tau^{(i)}$, the process model is given by

$$\tilde{\mathbf{t}}_\tau^{(i)} = h\left(\mathbf{t}_{\tau-1}, \theta_\tau^{(i)}, \Delta\tau, \Lambda, n_\tau\right), \tag{4.5}$$

[3] Although from a probabilistic point of view, variables s and d are deterministic and should not be defined in the state, for completeness of agent state representation, we have included it in the state.

which can be written as

$$\begin{bmatrix} \lambda_\tau \\ x_\tau \\ y_\tau \\ s_\tau \\ d_\tau \end{bmatrix} = \begin{bmatrix} \lambda_{\tau-1} + \theta_\tau^{(i)} + n_\tau \\ x_{\tau-1} + \Delta\tau s_\tau \cos[\lambda_\tau] \\ y_{\tau-1} + \Delta\tau s_\tau \sin[\lambda_\tau] \\ s_{\tau-1} \\ d_{\tau-1} + \Delta\tau s_\tau \end{bmatrix}, \tag{4.6}$$

where $\Delta\tau$ is the time-step length, which is 0.5, mimicking human walk speed of two steps per second and n_τ is zero mean noise. Having proposed b next positions along b movement rays in view area Figure 4.3, we need to compute the acceptance probability of each with respect to agent strategies and behaviours.

4.4.3.2 Acceptance Probability

With A_τ defining the acceptance event, $p(A_\tau)$ models the acceptance probability for a move. Acceptance probability is determined considering the traversibility of the search area with respect to the three environment data sets,

$$A_\tau = A_\tau^{(E)} \cap A_\tau^{(V)} \cap A_\tau^{(T)}. \tag{4.7}$$

Although, effects of environment data sets on the agent movement can be interrelate and very complex to model, we assume, for simplicity, that they are conditionally independent. Therefore, the acceptance probability can be decomposed into

$$\begin{aligned} p\left(A_\tau|\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, \Lambda\right) =& p\left(A_\tau^{(E)}|\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, \Lambda\right) p\left(A_\tau^{(V)}|\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, \Lambda\right) \\ &\times p\left(A_\tau^{(T)}|\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, \Lambda\right) \end{aligned}, \tag{4.8}$$

which can be written as

$$\begin{aligned} p\left(A_\tau|\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, \Lambda\right) =& p\left(A_\tau^{(E)}|\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, P, E\right) p\left(A_\tau^{(V)}|\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, P, V\right) \\ &\times p\left(A_\tau^{(T)}|\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, P, T\right) \end{aligned}. \tag{4.9}$$

To compute the acceptance probability, we need to compute and assign a measure to each term on the right hand side of (4.9).

Since we are modelling only local interactions, we will refer to the resultant agent's movement as *local movement* from here on. To compute the local movement, we need information about current cell the agent is in i.e. $\mathbf{t}_{\tau-1} \in a_c$ and the neighbouring cell a_n the agent would get to if continued to travel along proposed next position $\tilde{\mathbf{t}}_\tau^{(i)}$. Subscripts c and n stand for current

and neighbouring respectfully.

Using the information about a_c and a_n, the acceptance probabilities are set up based on adjacent types,

$$\begin{aligned} p\left(A_\tau^{(V)}|\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, P, V\right) &= \mathbf{M}^V_{(V(a_c),V(a_n))} \\ p\left(A_\tau^{(T)}|\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, P, T\right) &= \mathbf{M}^T_{(T(a_c),T(a_n))}. \\ p\left(A_\tau^{(E)}|\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, P, E\right) &= \mathbf{M}^E_{(E(a_c),E(a_n))} \end{aligned} \tag{4.10}$$

same as (2.21) to (2.16).

Having presented the local agent model, we now describe the process of initial distribution generation using it .

4.4.4 Initial Distribution Generation using Agent Particles

The pseudo-code is shown in Algorithm (4). After specifying and setting the environment data

Algorithm 4 Agent-based initial distribution generation

$\mathbf{AgentBasedInitialDistributionGeneration}()$
Set environment
Set LP Behaviour parameters
Initialise agent particle States: $\mathbf{t}_1^{(i:N)} \sim p(\mathbf{t}_L|P)$
for $i = 1 : N$ **do**
  Sample topography transition matrix: $\mathbf{M}^T \sim \mathcal{G}(\mathbf{M}^T_\mu, \mathbf{M}^T_\sigma)$
  Sample Vegetation transition matrix: $\mathbf{M}^V \sim \mathcal{G}(\mathbf{M}^V_\mu, \mathbf{M}^V_\sigma)$
  Sample Elevation transition matrix: $\mathbf{M}^E \sim \mathcal{G}(\mathbf{M}^E_\mu, \mathbf{M}^E_\sigma)$
  Set initial ϑ_L and l_L configuration.
  $\tau = 1$
  while run **do**
    $ViewArea = DetermineViewArea\left(\mathbf{t}^{(i)}_{\tau-1}, \vartheta_L, l_\tau, E, T, V\right)$
    Propose next position using a kinematic model of LP movement.
    for $j = 1 : b$ **do**
      $\tilde{\mathbf{t}}_\tau^{(i,j)} = h\left(\mathbf{t}_{\tau-1}, \theta_\tau^{(i)}, \Delta\tau, \Lambda, n_\tau\right)$
    end for
    Let $\mathbf{M} = \{\mathbf{M}^E, \mathbf{M}^T, \mathbf{M}^V\}$
    Compute acceptance probability for each proposed move considering environment data sets and transition matrices
    for $j = 1 : b$ **do**
      $A_\tau^{(j)} = ComputeMoveAcceptance\left(\mathbf{t}^{(i)}_{\tau-1}, \tilde{\mathbf{t}}^{(i,j)}_\tau, \Lambda, \mathbf{M}, ViewArea\right)$
    end for
    Select the next position: $\mathbf{t}_\tau^{(i)} = \arg\max_{\theta_\tau}\{\tilde{\mathbf{t}}_\tau^{(i,j)}, A_\tau^{(j)}\}_{j=1}^b$
    Set $run = false$ if the trail should be terminated
  end while
end for

sets, the LP behaviour is initialised by specifying distributions over the transition parameters in

Table 2.3. The state of each agent is then initialised at the PLS with samples drawn from the distribution specified over the reported last bearing and speed of the LP.

The agent particles are propagated sequentially, performing the following steps at every time-step:

- Step 1: Each agent particle is given a unique set of behaviours by sampling from the specified transition matrices in Table 2.3.
- Step 2: The agent's current state is propagated forward in time from $\tau - 1$ to τ along the movement rays using the kinematics model (4.6), generating b proposed next points $\tilde{\mathbf{t}}_{\tau}^{(1:b)}$.
- Step 3: The agent's vision or view area is determined by identifying visible cells from agent's current location and bearing.
- Step 4: The move acceptance is computed for each of the proposed next positions by (4.9) considering the view area. If a_n is not in view area, the acceptance probability is set to a very low value. Acceptability terms are computed sequentially for all proposed poses and are normalised.
- Step 5: Then $\mathbf{t}_{\tau}$ is acquired by selecting the proposed next position that maximises agent's state transition likelihood.
- Step 6: Decide whether to terminate the trail or continue. Each agent keeps track of the distance travelled. Defining d_{max} to be the maximum distance the LP could walk, and because this is a basic model, the agent trail is only terminated when $d_{\tau} > d_{max}$. Not considering fatigue or stay put strategy.

  When $run = false$, the trail is terminated and next particle starts traversing the environment from PLS. However, if $d_{\tau} < d_{max}$, $run = true$, and the agent moves back to Step 2 and continues with steps 3,4,5 and 6.

Having detailed the local agent model and the behaviours encoded, we need to verify that the behaviour encoded produce expected results.

4.5 Local Agent Model Behaviour Evaluation

The purpose of this section is to verify the key behaviours encoded into the local agent model.

This and all the other experiments presented in this thesis are implemented using MATLAB R2012b and carried out using an XPS L421X Dell Machine with Itel(R) Core (TM) i7-3667U CPU at 2GHz with 8 GB RAM running Windows 7 Professional.

Table 4.1: Parameters used in the mechanics of the basic agent model. Agent speed is specified based on [73].

Parameter	b	$\mu_{s_{PLS}}$	$\sigma_{s_{PLS}}$	$\Delta\tau$	σ_n	$\mu_{\lambda_{PLS}}$	$\sigma_{\lambda_{PLS}}$
Value	5	$1.3ms^{-1}$	$0.5ms^{-1}$	0.5	$0.05°$	λ_{PLS}	$10°$

4.5.1 Environment Setup

The experiments are performed using Sand Dunes data sets, presented in Section 2.5.2.

4.5.2 Agent Setup

The agent-based initial distribution is generated using $N = 50$ agent particles, each iteratively propagated according to Algorithm (4).

The model uses the transition parameter values specified in Table 2.3 unless stated otherwise. Other parameters used for the mechanics of the agent model are listed in Table 4.1.

Considering $d_{max} = 1950m$ and average walking speed of $1.5ms^{-1}$ [131], and assuming that on average two steps are taken per second, the agent model is run for 2600 time steps with each time-step representing a step taken by the LP.

4.5.3 Evaluation Results

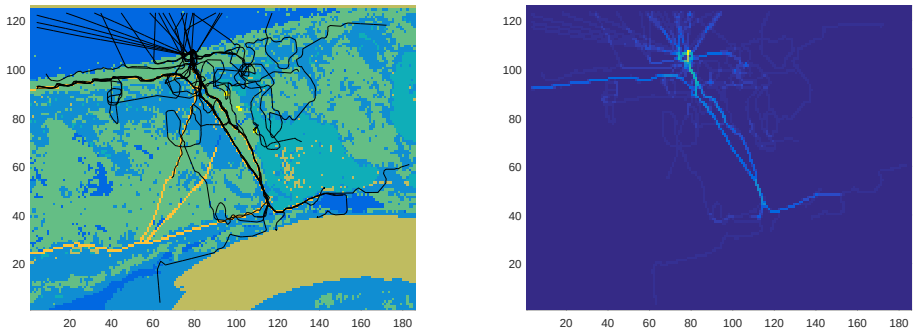
Agent strategies path travelling, random travelling and view enhancing are encoded into the transition matrices (2.15), (2.16) and (2.17), which are acquired by sampling from distributions specified in Table 2.3. The agent model can be made to prefer one strategy over another by setting the transition values in matrices in Table 2.3 accordingly.

4.5.3.1 Path Travelling Strategy Preference

The agent path travelling strategy is modelled in the topography transition matrix (2.16). To excite this strategy, we set all transition to path terms in $\mathbf{M}_\mu^T$ (the fourth column), to high values,

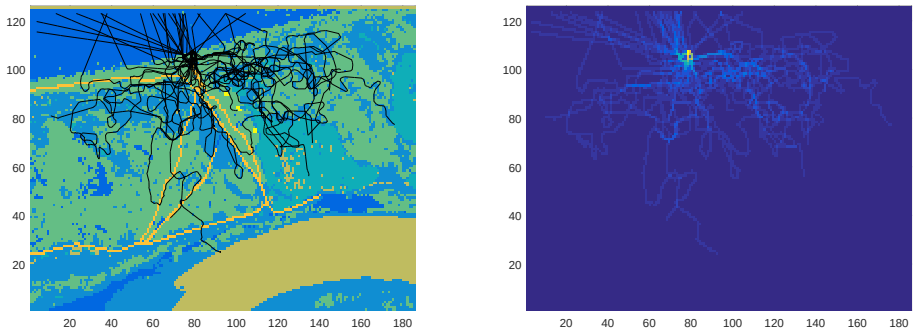
$$\mathbf{M}_\mu^T = \begin{bmatrix} 0.10 & 0.10 & 0.25 & 0.55 \\ 0.10 & 0.10 & 0.25 & 0.55 \\ 0.05 & 0.05 & 0.25 & 0.65 \\ 0.05 & 0.05 & 0.25 & 0.65 \end{bmatrix}.$$

With this setting, we get the path following behaviour illustrated in Figure 4.4. As can be seen, most of the agents follow the path. There are few agents that seem to be moving randomly. This however is due to the local interaction of the agent model. When faced with untraversable regions, agents make sudden change in their direction, as a result of which, they do not get close enough to path features to consider them. Having said this, from the distribution over the trail Figure 4.4b, computed using (3.12), we can clearly identify potential paths traversed by the LP.



(a) Agent trails **(b)** $p(a_i \in \mathbf{T}_B)$

Figure 4.4: Local agent model path travelling strategy.



(a) Agent trails **(b)** $p(a_i \in \mathbf{T}_B)$

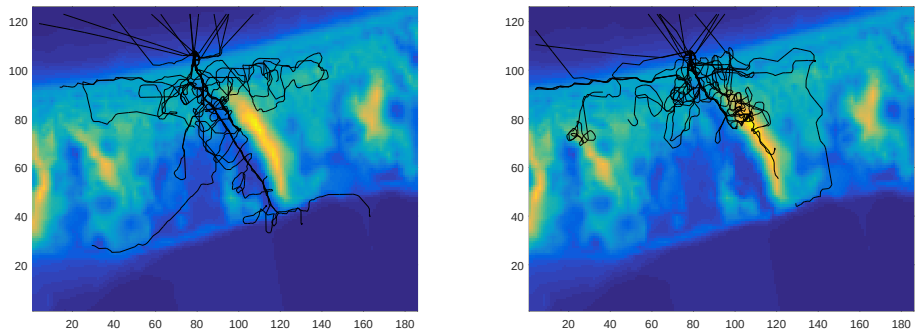
Figure 4.5: Local agent model random travelling strategy.

4.5.3.2 Agent Random Travelling Strategy

This strategy is the result of following the path of least resistance. It is modelled by mix of transition terms in all three transition matrices. It can be exited in different ways. For example, if no paths existed, the previous example would have also resulted in random agent movement. To show the agents capability to perform random travelling in the presence of paths, we set all transition to ground terms in $\mathbf{M}_{\mu}^{T}$ (the third column) higher than transition to path terms.

$$\mathbf{M}_{\mu}^{T} = \begin{bmatrix} 0.10 & 0.10 & 0.55 & 0.25 \\ 0.10 & 0.10 & 0.55 & 0.25 \\ 0.05 & 0.05 & 0.65 & 0.25 \\ 0.05 & 0.05 & 0.65 & 0.25 \end{bmatrix}$$

This makes agents traverse areas that offer least resistant and ignore natural path features, resulting in random movement illustrated in Figure 4.5.



(a) Downhill preference **(b)** Uphill preference

Figure 4.6: Agents behaviour of moving uphill to higher ground or staying / moving downhill in lower elevated ground. The downhill and uphill results are overlaid over the DTM model of the area.

4.5.3.3 Agent View Enhancing Strategy

Agent's behaviour of moving downhill or uphill also called view enhancing is encoded in elevation transition matrix (2.17).

Again, to show agent's capability to perform or avoid view enhancing, we perform two experiments: In the first experiment, we manipulate $\mathbf{M}_{\mu}^{E}$ specified in Table 2.3, and set downhill preference transition terms to high values,

$$\mathbf{M}_{\mu}^{E} = \begin{bmatrix} 0.2 & 0.2 & 0.45 & 0.149 & 0.001 \end{bmatrix}. \tag{4.11}$$

In the second experiment, we manipulate $\mathbf{M}_{\mu}^{E}$, and set uphill preference transition terms to high values,

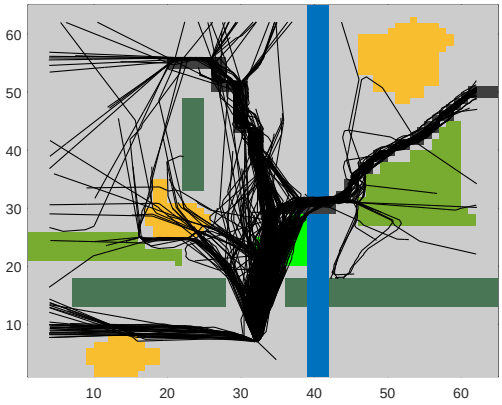
$$\mathbf{M}_{\mu}^{E} = \begin{bmatrix} 0.2 & 0.2 & 0.149 & 0.45 & 0.001 \end{bmatrix}. \tag{4.12}$$

The resultant behaviour is illustrated in Figure 4.6. With downhill movement preference illustrated in Figure 4.6a, we can see that agents stay in low elevated areas. In contrast, with uphill movement preference illustrated in Figure 4.6b, most agents move to higher ground and stay there.

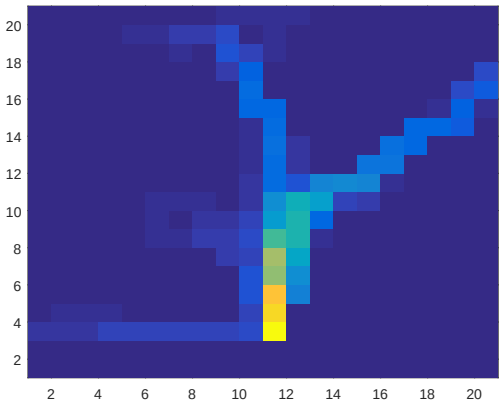
4.5.4 Importance of Keeping Agent Trail

To illustrate the importance of keeping history of agent movements – agent's trail, we revisit the bridge experiment detailed in Section 2.7.1. The results are presented in Figure 4.7. Figure 4.7a shows the agent trail samples, and Figure 4.7b shows the initial distribution computed from the ensemble of agent trails.

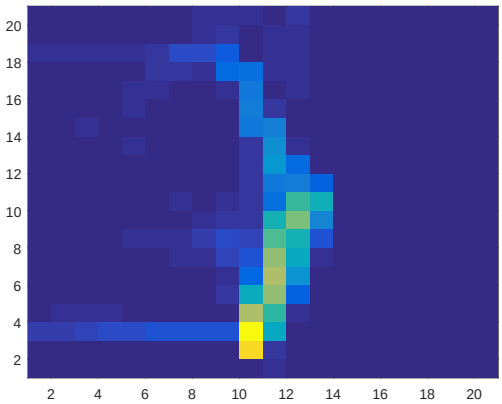
When the bridge is not detected as it no longer exists illustrated in Figure 2.15b, this



(a) Agent trails



(b) $p(a_i \in \mathbf{T}_B)$



(c) $p(a_i \in \mathbf{T})$

Figure 4.7: Bridge scenario with agent trail-based distribution. Figure 4.7a shows the agent trail samples. Figure 4.7b shows the generated trail-based initial distribution. Figure 4.7c shows the updated distribution as a result of new observation of the environment (land cover) - no bridge. The low resolution discretisation of the generated distributions represent search platform's camera frame size.

information is used to updated the weight on all agent trail particles. As a result weight on agent trail particles going through cell associated with bridge will go down and the weight on all other particles will increase. This results in an updated distribution illustrated in Figure 4.7c. As we can see, in contrast to diffusion-based distribution where the observation of water instead of bridge had no bearing on the overall distribution, here the update using the observation basically halves the search area. We will present implementation detail of the update model later in Chapter 7.

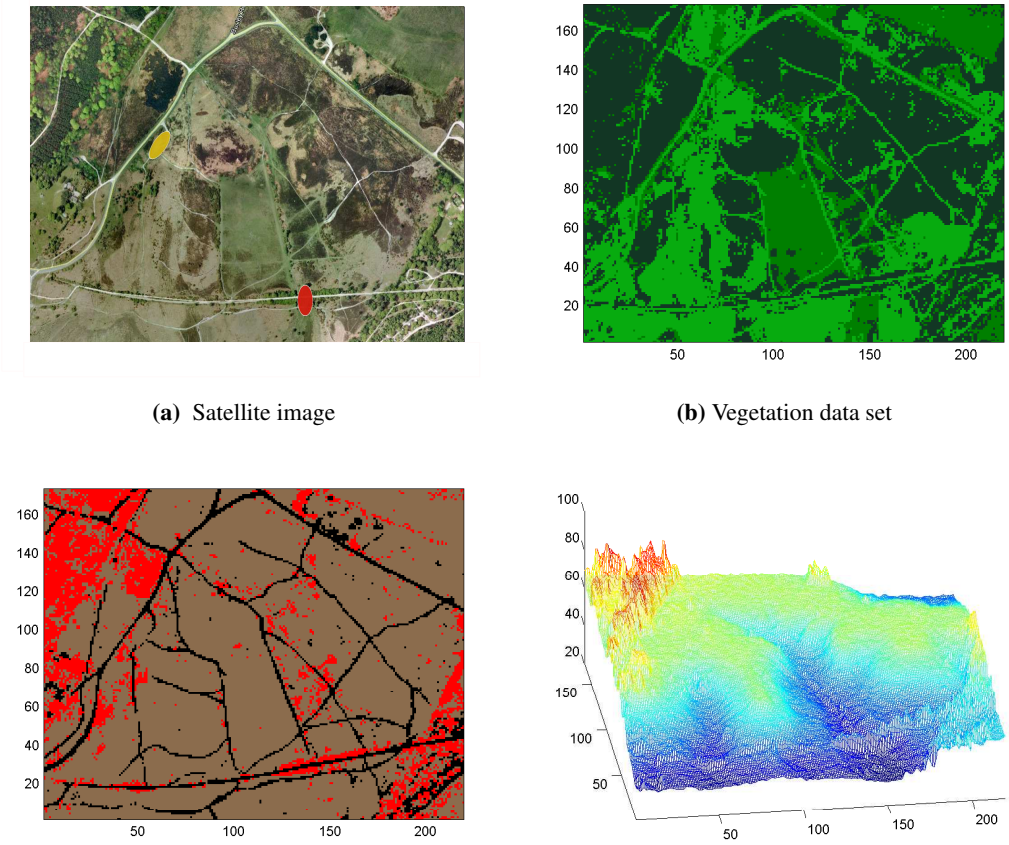
Having evaluated the behaviours encoded in the local agent model and demonstrated the significance of considering trails, we need to find out if the generated agent trails follow movement pattern of people in wilderness. One approach to this is to compare the generated distribution with logs of actual LPs's movements. However, this data is not typically available and had to be collected in a data collection experiment.

4.6 Data Collection

We planned to perform the data collection experiment over a large wilderness like area with rich diversity in vegetation, elevation and topography. The design of the experiment had to excite behaviours exhibited by lost hikers. However, for logistical and ethical reasons, the experiment had to be carried out in an area no larger than $1km \times 1km$, that could be controlled and posed no danger to the life of experiment participants. We identified part of the New Forest, UK, with coordinates $50°48'07.96''$ N $1°38'4.12''W$ illustrated in Figure 4.8, which met these requirements.

It has a rich diversity of vegetation, elevation and topography necessary to replicate wilderness. Figure 4.8a shows the start and end locations indicated by red and yellow circles respectively. The scenario required all participants to began at a common start location (which was treated as the PLS), located on one side of a bridge and traverse the terrain trying to locate a car park (representing safety and ultimate goal of the LP). Figure 4.9 shows some views of the search area from the participants point of view. The area was selected because it was challenging, and we believe helped excite various LP behaviours due to the following reasons:

- Participants were not familiar with the area, so they had no idea of the topography.
- The environment consisted of elevated areas that allowed participants move up hill to get a better view and locate the target car park.
- Participant's views were restricted by both vegetation and the uphill slope. This forced all participants to use their instinct and environment cues to get oriented and find the target



(a) Satellite image

(b) Vegetation data set

(c) Topography data set

(d) Elevation data set

Figure 4.8: Different representation of the search environment. In Figure 4.8a, the red and yellow ovals represent the PLS and end location. Figures 4.8b and 4.8c represents the classification of the search area satellite image in terms of vegetation and topography respectively. In Figure 4.8b, the variation in green colour represent the the three vegetation densities we are considering with light green representing sparse vegetation and dark green representing dense vegetation. In Figure 4.8c, the colours black, grey and red represent paths, obstacles, land cover classes of topology. Finally in Figure 4.8d the elevation model of the environment is shown in 3D with varying red to blue colours representing the highest to lowest elevation.

location.

- There were a number of trails, both over open grassy areas and in pathways that run between dense foliage. This information was used to assess participants path following behaviour.

- There were no tall obstructions that influenced the GPS signals. This was very important, especially when we had access to very small and cheap GPS units.

- The environment contained several car parks acting as places of interest, target or goal for the participants. Although in high elevated areas, these car parks were hidden by the vegetation and could not be seen from miles away.

(a) Point of view 1

(b) Point of view 2

(c) Point of view from PLS

Figure 4.9: View of the search area from data collection participants' point of view. Figure 4.9c represents a participant's view from PLS.

The experiment was carried out with the help of 12 participants. Each participant performed the task individually. During the experiment each participant was accompanied by the researcher who carried the GPS equipment (the size of which was substantial and would have altered the participants movements) and had very good knowledge of the area. This insured that the participants stayed within a confined area and were safe. Also, researcher walked behind them not to influence the participant's movements. A second static GPS was used as a base station to provide differential correction and errors within about 1.5m. The differential correction was performed using GravNav software running on a Windows operating system. The GPS logs, overlaid on Google Earth, are shown in Figure 4.10.

4.7 Local Agent Model Evaluation

To evaluate the local agent model, we will investigate and assess if agent generated trails follow pattern of GPS logs of the data collection experiment illustrated in Figure 4.10. Looking at this figure, we treat the base location used for data collection experiment (red circle) as the PLS and the car park (yellow circle) as ideal safety location the agent would want to get to.

4.7.1 Environment Setup

This experiment is performed using New Forest data sets illustrated in Figure 4.8.

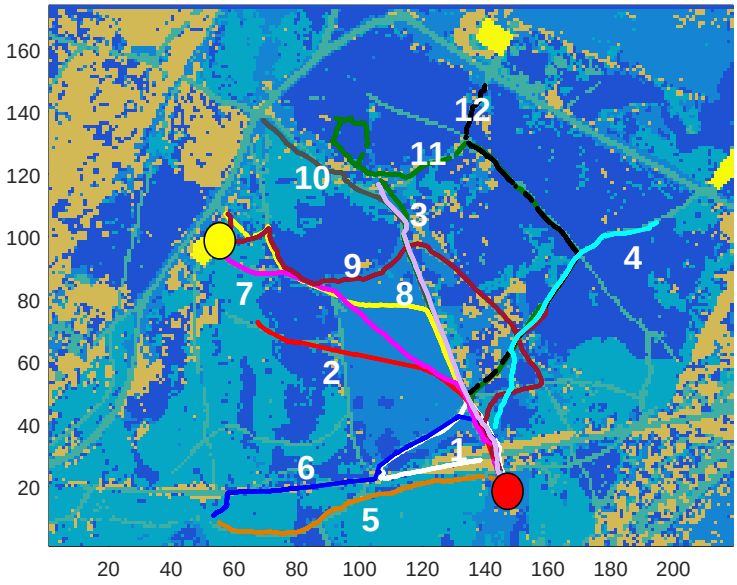
4.7.2 Agent Setup

The agent model setup for this experiment is similar to agent model setup in Section 4.5.2. The only difference is that in this experiment, the simulation is run for 1560 time-steps, which is equal to 13 minutes of walking and $d_{max} = 1170m$, the time and distance walked by data collection participants.

4.7.3 Evaluation Results

Figure 4.10a shows the first 13 minutes of the GPS logs of each participants. Qualitatively, these logs support the behaviour predicted in [3]: when lost people encounter linear features they will follow them until their goal is achieved or they are forced to abandon that feature. The traces also show that when a LP spots a destination globally, they will plan accordingly. For example, when the car park (the target location) indicated by yellow circle is observed, trails 7,8 and 9 suddenly turn towards it performing direction travelling strategy.

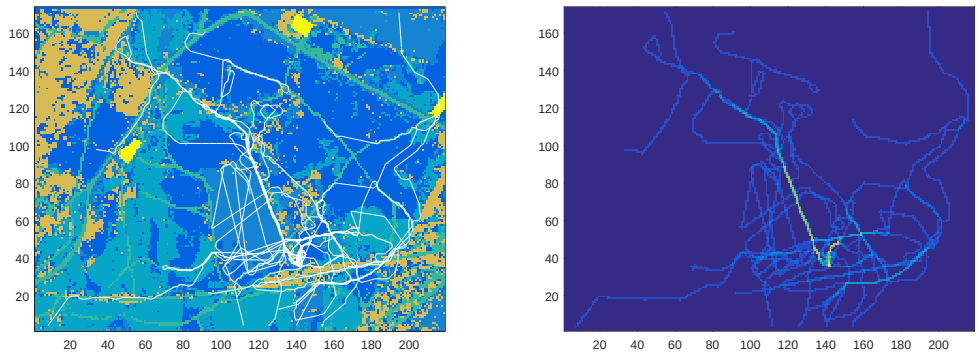
The agent trails are illustrated in Figure 4.11. As can be seen, the agents are able to follow linear features and features that offer least resistance reflecting the encoded behaviour. However, since they are not able to perform direction travelling, they are not able to take advantage of important features like the car park. For this reason, none of the agents change direction and



(a) GPS log of participants overlaid on the image of search area

(b) Global Positioning System logging equipment carried by a colleague during trial runs

Figure 4.10: Figure 4.10a shows the search area overlaid with 13 minute of logged GPS trails. In the experiment, participants started at the red circle (the PLS) and had the goal of reaching the yellow circle (the target location), the location of which was unknown to them. Figure 4.10b shows GPS logging equipment. In this picture, the equipment is carried by a colleague during trial runs.



(a) Agent trails **(b)** $p(a_i \in \mathbf{T}_B)$

Figure 4.11: Trail-based distribution using local agent model with only local interaction.

go towards the car park. Also, since only local observations are considered, the agents had to be in close proximity of features to consider them. This means, important environment cues such as paths were missed, which resulted in agent trails that are not representative of lost people movement. Other limitations of local agent generated trails are that:

- They are not smooth. They make sudden sharp turns when faced with low traversable regions.

- Their lengths are the same. This because, stay put strategy and agent's state depended behaviour such as fatigue, speed variation are not considered. Agents stop when $d_\tau > d_{max}$.

Since agent trails temporally connect different regions within the search area, during the search phase, it has the potential to reduce the search area globally based on observations made locally. However comparing the agent trails with logs of people movements, we show the local agent model is not able to faithfully represent lost people movement. This because, the local agent model lacks global interaction capability and some state dependent behaviours such as goal directed strategies, speed variations and fatigue. Therefore, the next obvious step is to model and incorporate these features into the agent model.

4.8 Summary

We started this chapter by reviewing the hiker movement-based agent models. We argued that current ABMs do not model the behaviour of actually being lost. They are trail oriented and make certain assumptions such as known location. Therefore, we proposed to design a new agent model capable of capturing LP behaviour. We presented this agent model and described

its implementation with local interaction capability. We evaluated the various behaviours encoded and then the trails generated, comparing the later with movement data of people, collected in a controlled experiment. We showed that compared with the diffusion generated distribution can help significantly reduce search times. In the next chapter, we extend the agent model and introduce global interaction capabilities along with some state dependent behaviours like speed variations, fatigue and goal depended strategies.

Chapter 5

Global Agent Model of Lost Person Movement

5.1 Introduction

In the previous chapter we presented the local agent model. In this chapter we extend the local model to include global interactions with the environment and consider effects of the environment on agent's state dependent behaviour – energy consumption and speed.

We start by introducing the global interactions in Section 5.2. In order to capture the global interactions, we make certain improvements to the local model, which includes introducing a more sophisticated perception model that consists of configurable vision and memory, modelling global influence of environment on agent trail in terms of attractiveness and repulsiveness, and including additional, stationary and clear intention based re-orientation strategies.

We evaluate the performance of the agent model for each of the new improvements in Section 5.3. Once we have a model that is capable of interacting and negotiating with environment at local and global level, we introduce additional very important state depended behaviour exhibited by lost people– speed variation and energy consumption in Section 5.4. Finally, we evaluate the performance of the agent model as a whole in Section 5.5. We compare the performance of the proposed agent model to that of a diffusion model and other initial distribution types used in search literature and, present our summary in Section 5.6.

Key terms used in this chapter

- **FOV**: It is the sweep angle over which features in the environment are observable to an agent from a location and bearing.

- **VCL**: This models the maximum distance at which a feature can be seen and considered in the agent movement.

5.2 Global Agent Model

In this section we build on the local agent model presented in Section 4.4. At the core of this improvement is the consideration of features that are located at a distance from the agent. To capture and consider features both near and far, agents requires a more sophisticated perception model.

5.2.1 Agent Perception

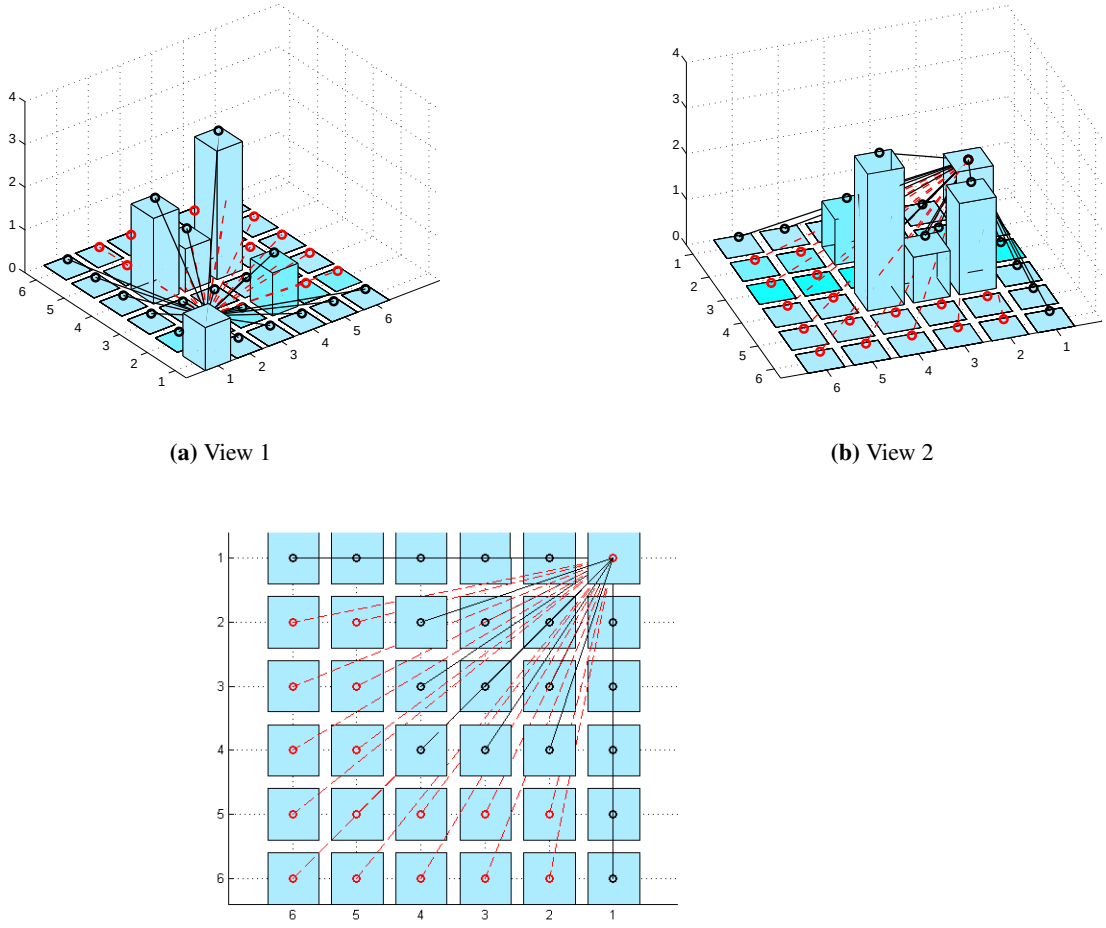
The hikers' decision making is strongly determined by what they can see [72, 73] i.e. missing a landmark or a view can result in getting lost and becoming disoriented [71]. Therefore, it is very important to model agent vision that considers environment occlusions and captures features located both near and far from the agent. In addition to the vision system, agents need to build some awareness of the search area by remembering information about the explored search area such as places of interest, recent places visited. As a result, agent's perceptual model should include both *vision*, which models an agent's ability to see both near and far objects, and some sort of *memory*.

5.2.1.1 Vision

We model environment occlusions by using a visibility graph [132, 133] of the search area using the Digital Surface Model (DSM). This is illustrated in Figure 5.1. To model realistic representation of LP vision, view area is parameterised by different configurations of FOV and VCL:

1. **FOV.** Because people do not always look in one direction, as they at times look around, the angle over which features can be perceived by an agent has two possible values: *narrow* (N^{FOV}) and *wide* (W^{FOV}). N^{FOV} is used when an agent moves forwards in a purposeful manner, for example when following a linear feature. W^{FOV} is used when the agent looks around. For example, an agent will search for an alternative directions of travel when at a junction or when the path forwards is blocked.

2. **Vision VCL.** An agent has two modes of operation: *close* (C^V) and *far* (F^V). Close vision is used when an agent checks locally for traversibility of the area close to it. In this mode, features are seen only if the distance to the agent is less than S^{VCL} (short visual distance). Far vision is used when the agent is looking around to, for example, reorient or select a new point to head towards. Some features (such as path and water features) can be seen at the medium distance M^{VCL}. Other features — such as houses — can be seen from an even greater distance L^{VCL}.



(a) View 1

(b) View 2

(c) View 3

Figure 5.1: Visibility determination using visibility graph constructed for an observer at cell(1,1). The three Figures illustrate the same visibility from different angles. The red lines and circles indicate the cells that cannot be observed. The black lines and circles indicated the cells that can be observed.

Different configurations of view area are illustrated in Figure 5.2.

Defining $\mathbf{V}_{a_c}$ to be the visibility graph of the search area from current cell a_c, with $\vartheta_{\tau-1}$ and $l_{\tau-1}$ representing the sweep angle and critical length of view area configuration determined in the previous time-step $\tau - 1$, the agent's view area is computed by

$$ViewArea = DetermineVewArea(\mathbf{V}_{a_c}, \vartheta_{\tau-1}, l_{\tau-1}, \lambda_{\tau-1}). \tag{5.1}$$

where $\lambda_{\tau-1}$ is the agent's bearing in the previous time-step. This is illustrated in Figure 5.3. Figure 5.3a shows the overall visibility of the area computed using MATLAB viewshed function given agent's current position, height and DSM, and figure 5.3b shows the view area given the agent's bearing, FOV, VCL and visibility of the view area. The affect of environment occlusions

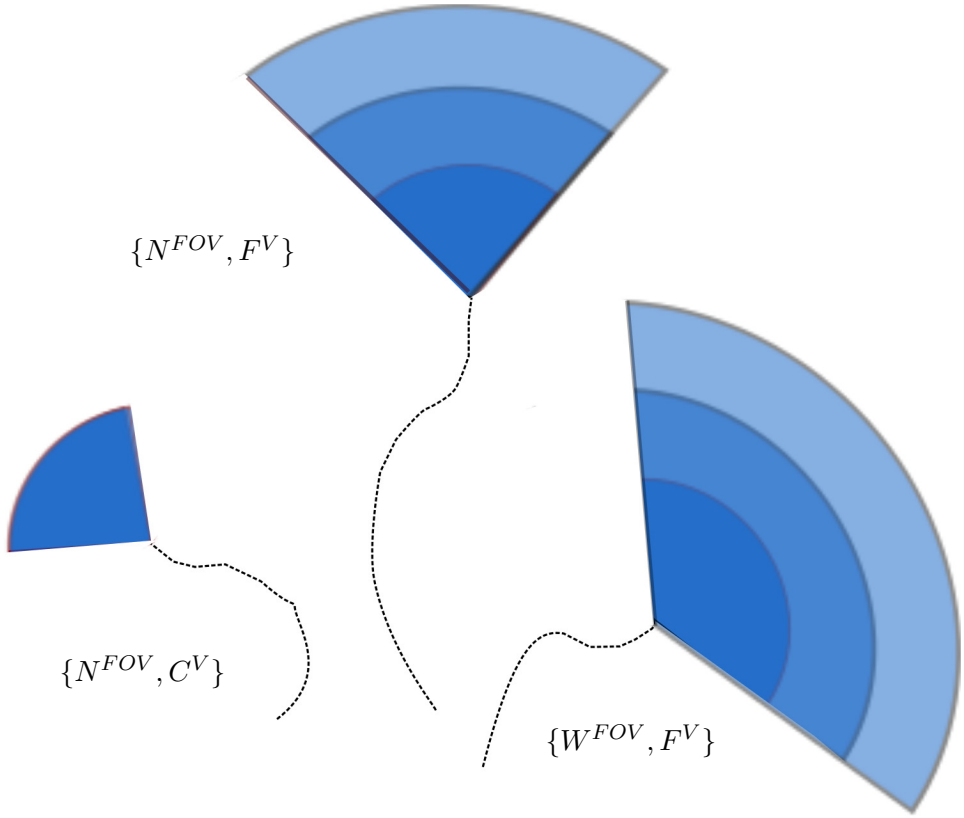
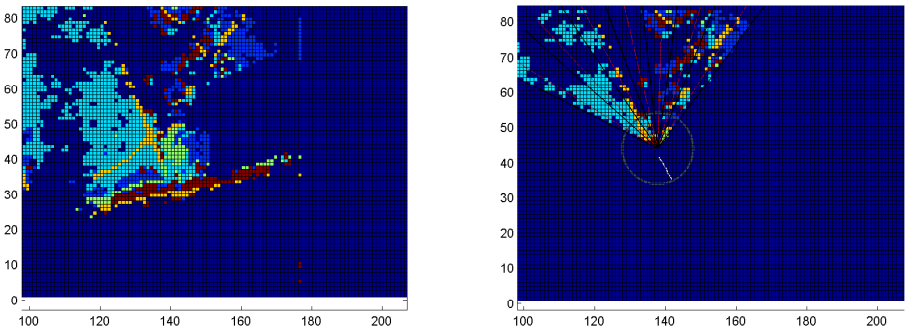


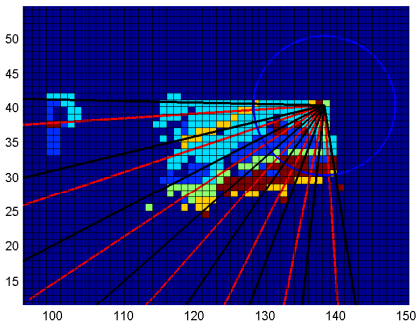
Figure 5.2: There are there view modes-looking narrow and near, looking narrow and far, and looking far and wide. The dark to light blue areas represent agents VCLs at close, medium and long ranges respectively.



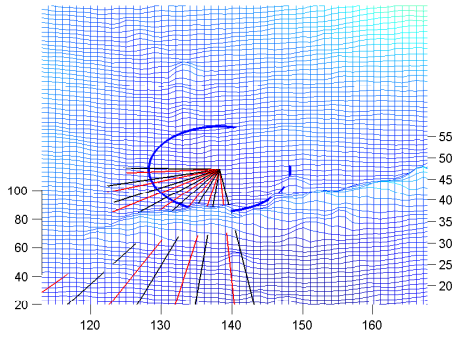
(a) Visibility of the area

(b) Agents view area

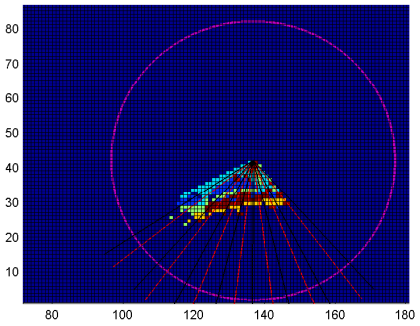
Figure 5.3: Determining agent's view area. Figure 5.3a shows the visibility of the area $\mathbf{V}_{a_c}$ given agent's current position, height and DSM. Figure 5.3b shows the agent's view area given agent's $\lambda_{\tau-1}$, $\vartheta_{\tau-1}$, $l_{\tau-1}$ and $\mathbf{V}_{a_c}$. The pixel colours represent the classified feature types.



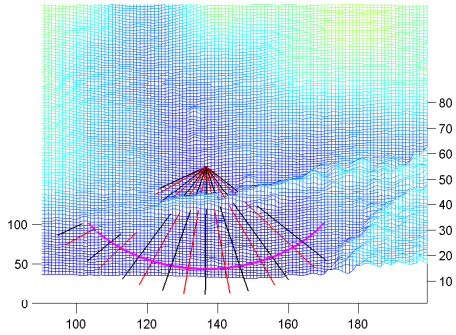
(a) Observable features with view configuration $(\{N^{FOV}, C^V\})$.



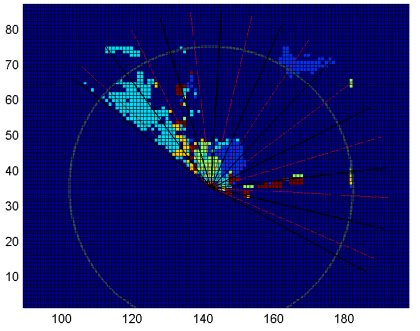
(b) Interaction with elevation.



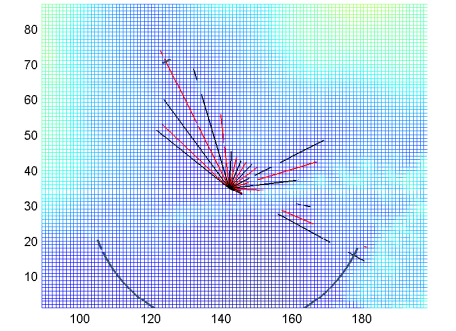
(c) Observable features with view configuration $(\{N^{FOV}, F^V\})$.



(d) Interaction with elevation.



(e) Observable features with view configuration $(\{W^{FOV}, F^V\})$.



(f) Interaction with elevation.

Figure 5.4: The agent's constrained visual field. The radius of the circle is used to model VCL. Agent vision is divided into bins defined by the black rays. The red coloured rays in each bin represent the central rays. Figures on top row show the narrow-close condition $(\{N^{FOV}, C^V\})$. Figures in the middle row show narrow-far condition $(\{N^{FOV}, F^V\})$ and figures in the bottom row show wide-far condition $(\{W^{FOV}, F^V\})$. The occluding effects of obstacles such as elevation are illustrated in the figures on the right column. The colours (excluding navy colour) in the observable feature Figures represent the observable classified cells.

on the agent's vision is illustrated in Figure 5.4. The most common setting for the view area is $(N^{FOV}, C^V\})$, which is used when an agent has a fixed objective and is monitoring the local

environment to check for obstacles and features [130]. However if the agent's attempt to reach its goal is stymied in some way, the agent can decide to look further forwards in its current direction using $(N^{FOV}, F^V\})$ or look around more generally using $(W^{FOV}, F^V\})$. Both of these actions make it possible for the agent to perceive global features in the environment. As a result, the agent can make early adjustments to its path moving towards or away from features and obstacles. Pseudo-code of selecting vision configuration is given in Algorithm (5). The

Algorithm 5 Computing agent view area configuration

ComputeViewAreaConfiguaration()
INPUTS: $isWideView, nextLongViewTime, \mathbf{t}_{\tau-1}, p\left(W^{FOV}\right)$
OUTPUTS: $nextLongViewTime, \vartheta_\tau, l_\tau$
if $\tau = 1$ **then**
  Set $l_\tau = F^V$
  if $rand > p\left(W^{FOV}\right)$ **then**
    Set $\vartheta_\tau = W^{FOV}$
  else
    Set $\vartheta_\tau = N^{FOV}$
  end if
else if $isWideView = true$ **then**
  Set $l_\tau = F^V$
  Set $\vartheta_\tau = W^{FOV}$
  Set $wideView = false$
else if $\tau < nextLongViewTime$ **then**
  Set $l_\tau = C^V$
  Set $\vartheta_\tau = N^{FOV}$
else if $\tau >= nextLongViewTime$ **then**
  Set $l_\tau = F^V$
  Set $nextLongViewTime = \tau + n : n \sim Pois\left(\mu^{nextLongViewTime}\right)$
  if $rand > p\left(W^{FOV}\right)$ **then**
    Set $\vartheta_\tau = W^{FOV}$
  else
    Set $\vartheta_\tau = N^{FOV}$
  end if
end if

configuration $(W^{FOV}, F^V\})$ is mainly used by setting $isWideView$ to $true$ when needed during agent movement. This is done when

- *The path forward is blocked*: The agent turns back and uses W^{FOV} to look for alternative directions or places of interest.
- *The agent reaches a junction*: Here, W^{FOV} is used to decide on path forward.
- *The agent reaches an intermediate goal*: Here, W^{FOV} is used to check for another goal or place of interest.

Table 5.1: Parameters used in the configuration of the agent vision. VCL is determined heuristically.

Parameter	S^{VCL}	M^{VCL}	L^{VCL}	N^{FOV}	W^{FOV}	$p(W^{FOV})$	$\mu^{nextLongViewTime}$
Value	$50m$	$200m$	$300m$	$100°$	$170°$	0.6	10

- *The agent decides to look around randomly*: The agent may decide at random to use W^{FOV} to look for places of interest.

Depending on the FOV configuration of view area, for *propose–accept* computation, inspired by [72, 129], we discretise the FOV at different regular intervals, each with a central ray. For N^{FOV}, with respect to $\lambda_{\tau-1}$, these central ray (movement ray) are at

$$\theta_\tau = \begin{bmatrix} -35° & -21° & -7° & 7° & 21° & 35° \end{bmatrix},$$

and for W^{FOV}, these central ray (movement ray) are at

$$\theta_\tau = \begin{bmatrix} -70° & -50° & -30° & -10° & 10° & 30° & 50° & 70° \end{bmatrix}.$$

Table 5.1 presents the vision parameters along with the values specified in this research.

5.2.1.2 Memory

Each agent possess a rudimentary memory system. This memory system stores the agent's understanding of the environment. It is used to formulate strategies and execute agent motion, particularly when performing various re-orientation strategies [74]. We support both *short-term* and *long-term* memory.

1. **Short-term memory,** M_S: This type of memory keeps track of recent movements. This memory type has two sub-types. The first consists of recent history of locations visited by the agent, $M_{S1,\tau} = \{\mathbf{t}_\tau, \cdots, \mathbf{t}_{\tau-m}\}$. This is used to help the agent back track when needed or, conversely, avoid unintentional doubling back. The second, consists of the recent history of the heading of the agent $M_{S2,\tau} = \{\lambda_\tau, \lambda_{\tau-1}, \cdots, \lambda_{\tau-n}\}$. These are used to smooth changes in the direction of the agent's movement.

2. **Long-term memory,** M_L. This stores the recalled location of significant landmark features such as a house that an agent has recently noticed. However, the memory must account for the fact that the agent will become more uncertain of each landmark's location over time and can eventually be forgotten if not re-observed.

M_{S1} and M_{S2} are implemented using circular buffers. M_L consists of a weighted set of q landmark features,

$$M_{L,\tau} = \{\mathbf{f}_\tau^{(i)}, \omega_\tau^{(f,i)} : i = 1, 2, \cdots, q\}, \tag{5.2}$$

where $\mathbf{f}^{(i)}$ is the i^{th} feature, and is described using the tuple

$$\mathbf{f}_\tau^{(i)} = \left[q^{(i)}, \lambda^{(f,i)}, \varrho^{(i)}, \tau^{(f,i)}\right]_\tau, \tag{5.3}$$

where $q^{(i)}$ is the features id, $\lambda_\tau^{(f,i)}$ is the bearing to the feature at time τ, $\tau^{(f,i)}$ is the time the feature was last seen and $\varrho^{(i)}$ is the importance of feature type. If the feature was observed at time step τ, $\tau^{(f,i)} = \tau$. The weight of a feature is used to represent its importance when agent wants to select a goal target. Its value is computed by

$$\omega_\tau^{(f,i)} = \nu_\tau \ \varrho^{(i)}, \tag{5.4}$$

where ν_τ is used to weight observed features more than un-observed features. The memory update process is given in Algorithm (6).

At each time-step, all the visible features are detected and the bearing to them with respect to agent's current position is computed. Two actions are taken: First, features not in agent's memory are inserted $\mathbf{f}^{((i))} \rightarrow M_{L,\tau}$. Second, features that are already observed and are in the agent's memory are updated.

For those features in the memory that are not observed, two effects are modelled. First, the agent gradually loses knowledge of where the feature is. This is achieved by adding noise to the recalled heading of the feature, to model the fact that the agent becomes less certain about the spatial position of the feature. its value is given by.

$$\lambda_\tau^{(f,i)} = \lambda_{\tau-1}^{(f,i)} + n^f, \tag{5.5}$$

where n^f is zero mean noise with σ^f standard deviation. The second is that the agent can forget the feature $\mathbf{f}^{((i))}$ if $\tau > \tau^{(f,i)} + \tau_{max}^{\mathbf{f}}$ where $\tau_{max}^{\mathbf{f}}$ is the maximum time an unseen feature is retained in memory. If this time is exceeded, the feature is deleted from the memory.

5.2.2 Agent Strategies

In addition to the strategies encoded in the transition matrices (2.15), (2.16) and (2.17), we need to model the strategies that require agents to make a decision which include stay put, direction travelling, and back tracking. Considering the improved perception capabilities of the global

Algorithm 6 Memory update process

$\mathbf{MemoryUpdate}()$
if New features are observed **then**
  for $i = 1$: Number of features **do**
    $\lambda^{(f,i)} = \tan^{-1}\left(\frac{y_\tau - y^{f^{(i)}}}{x_\tau - x^{f^{(i)}}}\right)$
    if $\mathbf{f}^{((i))} \in M_{L,\tau}$ **then**
      $\lambda_\tau^{(f,i)} = \lambda^{(f,i)}$
      $\omega_\tau^{(f,i)} = \nu_\tau \ \varrho^{(i)}$
      $\tau^{(f,i)} = \tau$
    else
      $\omega_\tau^{(f,i)} = \nu_\tau \ \varrho^{(i)}$
      $\tau^{(f,i)} = \tau$
      $\mathbf{f}_\tau^{(i)} \rightarrow M_{L,\tau}$
    end if
  end for
else if Features in memory are not observed **then**
  for $i = 1$: Number of features not observed **do**
    if $\tau < \tau^{(f,i)} + \tau_{max}^{\mathbf{f}}$ **then**
      $\lambda_\tau^{(f,i)} = \lambda_{\tau-1}^{(f,i)} + n^f$
      $\omega_\tau^{(f,i)} = \nu_\tau \ \varrho^{(i)}$
    else if $\tau > \tau^{(f,i)} + \tau_{max}^{\mathbf{f}}$ **then**
      $\mathbf{f}_\tau^{(i)} \leftarrow M_{L,\tau}$
    end if
  end for
end if

Table 5.2: Agent memory related parameters.

Parameter	Symbol	Value
n^f standard deviation	σ^f	0.04
Tolerable M_L maximum time	$\tau_{max}^{\mathbf{f}}$	300 steps
M_{S2} history size	n	4

agent model, these strategies are performed in the following way.

- *Stay Put*: This strategy is chosen under two conditions: when the agent is suffering from fatigue or it randomly decides to not move and stay put. When an agent decides to stay put, its trail is terminated. Hence tracks of different length can be modelled. The decision to stay put is checked every p steps.

- *Direction Travelling*: This strategy is used by agents when they decide to go towards a target location or place of interest held in M_L. The agents continue with this strategy to get to the place of interest I (treated as goal g_τ) unless their way forward is blocked.

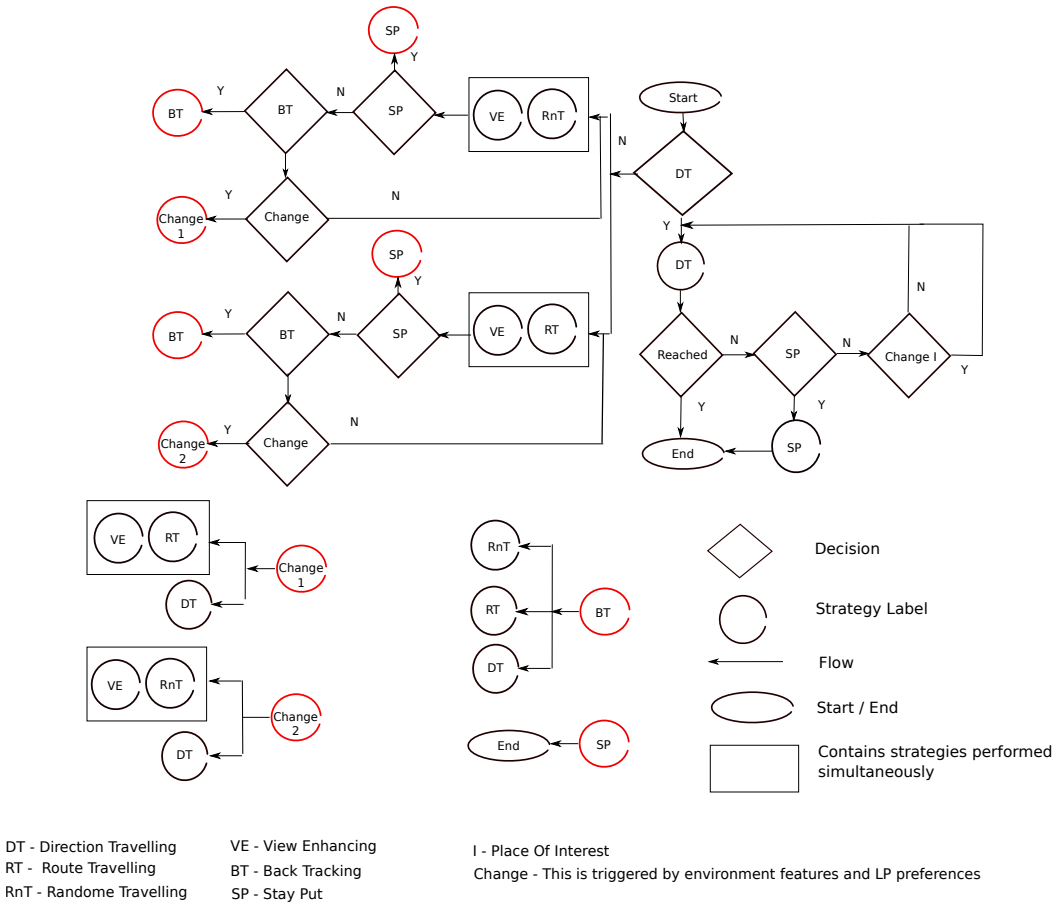


Figure 5.5: Agent strategy change flow diagram. The black circles represent the strategy label. The red circles are used to reference different part of the flow diagram. The diamonds represent decisions.

- *Backtracking*: Backtracking is performed using the short term memory of places stored in M_{S1}. This gives the agent opportunity to explore unseen areas and if a place of interest is observed, it can decide to stop backtracking, and instead, head towards the place of interest.

5.2.3 Choosing a Strategy

The flow diagram in Figure 5.5 shows how strategies are prioritised and executed. The ultimate aim of a LP is to get to a safe location. If a place of interest is visible to the agent, it may decide to perform direction travelling strategy to get to it. While doing this, the agent is free to change its mind and decide to go towards a different place of interest. However, if no place of interest is observed or the agent did not want to go to the place interest seen, it uses mix of random travelling or route travelling with view enhancing strategies depending on environment configuration. It is normal to perform two of these strategies simultaneously i.e. view enhancing with route travelling. The preference of route travelling over random travelling and preference of view enhancing depends on the encoded values in the transition matrices (2.15), (2.16) and (2.17).

Table 5.3: Strategy transition parameter values for a lost hiker.

Parameter	p	$p(SP)$	$p(DT)$	$p(ChangeI)$
Value	300	0.2	0.7	0.4

While performing route travelling or random travelling, depending on interaction with environment, the agent can decide to stay put, back track or decide to change strategy to another strategy. When backtracking, it can continue using any one of the three strategies: random travelling, route travelling, or direction travelling. The later if a place of interest is seen.

The decision value used to transition between strategies are specified in Table 5.3.

5.2.4 Agent Movement

5.2.4.1 Agent State

With changes in agent perception, the agent state in (4.3) is extended to

$$\mathbf{t}_\tau = \left\{ \begin{bmatrix} x & y & \lambda & g & s & d \end{bmatrix}^T \; M_{S1} \; M_{S2} \; M_L \right\}_\tau, \tag{5.6}$$

where g_τ is a discrete value which shows current goal, $M_{S1,\tau}$, $M_{S2,\tau}$ and $M_{L,\tau}$ are the short and long term memories. At the start of simulation, the agents states are initialised as

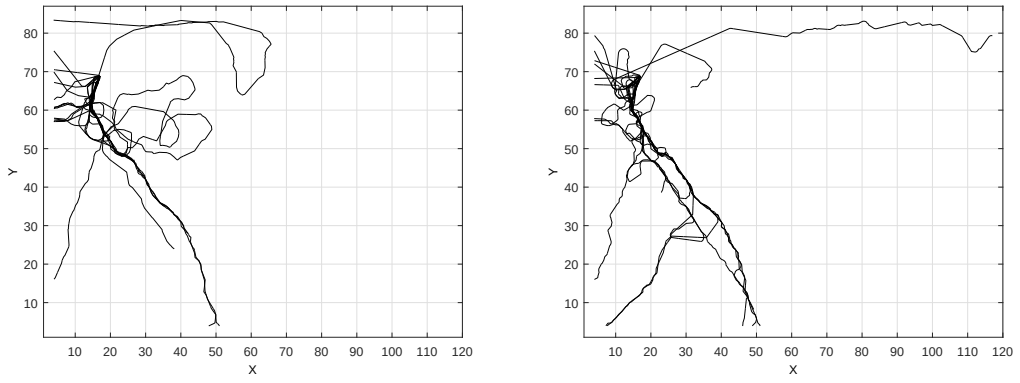
$$\mathbf{t}_L = \left\{ \begin{bmatrix} x & y & \lambda & g & s & 0 \end{bmatrix}^T \; M_{S1} \; M_{S2} \; M_L \right\}_L,$$

where $M_{S1,L}$, $M_{S2,L}$ and $M_{L,L}$ are all empty. Using this extended state, the agent operates iteratively traversing the environment using (4.6). Using $M_{S2,L}$, we model λ_τ to ensure smooth manoeuvring of agent movements. This is achieved by

$$\lambda_\tau = \mathcal{G}\left(\frac{1}{n} \sum_{i=1}^{n} \lambda_{\tau-i}, \sigma_\lambda^2 \right) + \theta_\tau^{(i)}, \tag{5.7}$$

where the first term on the right is derived from the agent's bearing history held in short term memory $M_{S2,L}$ holding n past bearings, with σ_λ modelling the perturbation of the bearing reflecting an LP's behaviour of deviation from a straight trail.

The effect of considering short term memory in computing λ_τ and as a result on the movement of the agent is illustrated in Figure 5.6, which is based on SandDunes data sets. Figure 5.6a shows results of agent movements when short term memory is not considered i.e. $n = 1$. As a result, and slowly, the agents tends to move in tight circular trajectories and are not able to explore the search area. This is specifically the case when faced with regions that offer some level of impediment, agents make sudden change in their heading trying to avoid those areas.



(a) $n = 1$. **(b)** $n = 4$.

Figure 5.6: Effect of considering agent's bearing history. The figure on the left shows 20 agent trail samples with $n = 1$ in computing the agent's heading λ_τ. The figure on the right on the other hand shows 20 agent trail samples with $n = 4$ in computing the agent's heading λ_τ.

Figure 5.6b on the other hand shows agents movements when short term memory is considered in determining new heading. As can be seen, this results in straighter agent trails with better coverage of the search area. This because it models agents general heading, which helps them avoids sudden changes in bearing (with large angles).

5.2.4.2 Acceptance Probability

Having covered the local interaction of the agent with environment in Section 4.4.3.2, in this section we model the effects of non-local features on the agent's movement. We will call this *global orientation*. One of the key features that affect global orientation is the places of interest at a distance requiring the agent to perform direction travelling. To model this, we introduce an additional acceptance event term $A_\tau^{(g_\tau)}$ to (4.9),

$$p\left(A_\tau|\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, \Lambda\right) = p\left(A_\tau^{(g_\tau)}|\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, P\right) p\left(A_\tau^{(E)}|\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, P, E\right) \times p\left(A_\tau^{(V)}|\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, P, V\right) p\left(A_\tau^{(T)}|\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, P, T\right). \quad (5.8)$$

Traversibility with Respect to a Target Location: When a move is proposed, the term $p\left(A_\tau^{(g_\tau)}|\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, P\right)$ determines the probability that $\tilde{\mathbf{t}}_\tau$ is compatible with the agent's desire to get to a target location or goal. We model this by

$$p\left(A_\tau^{(g_\tau)}|\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, P\right) = \frac{1}{\sqrt{2\pi\sigma_\lambda}} \exp\left\{-\frac{(|\lambda_\tau - \lambda^\mu|)^2}{2\sigma_\lambda}\right\}, \quad (5.9)$$

where λ_τ is the bearing of the proposed position and λ^μ is the mean bearing with respect to which the agent moves. To ensure the agent moves towards its intended target, (5.9) penalises

$\tilde{\mathbf{t}}_\tau^{(i)}$ for the amount of change in its bearing with respect to the mean bearing λ_μ. When the agent intention is to moves towards a definite goal, λ^μ is set to the bearing of the destination or goal feature $\lambda_\tau^{(f,i)}$ the agent wants to get to.

Traversibility with Respect to Global Elevation: We model the affect of slope on the agent's global orientation by computing the slope between the cell the agent is in, $\mathbf{t}_{\tau-1} \in a_c$, and a cell located at the end of movement ray λ_τ along the proposed next position $\tilde{\mathbf{t}}_\tau^{(i)}$. Denoting a_f to be this cell. The influence of the slope on agent's *global orientation* is computed by acquiring a transition value corresponding to the slope from (2.17) similar to (2.23). Therefore (4.10) is extended to

$$p\left(A_\tau^{(E)}|\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, P, E\right) = \mathbf{M}^E_{(E(a_c),E(a_n))}\mathbf{M}^E_{\left(E(a_c),E(a_f)\right)}. \tag{5.10}$$

Traversibility with Respect to Topography: In addition to local affects of topography on LP movements, there are certain topography features that can influence the LP's movement from very far, these include obstacles, water and paths have a significant global impact. Lost people normally tends to keep a lookout for features that can either attract or prevent them from continuing along a particular direction.

The influence of topography is computed by considering both the *repulsiveness* of obstacles like rivers; and the *attractiveness* of the linear features such as natural paths. Defining the repulsiveness and attractiveness to be R and A respectively, we can extend (4.10) to

$$p\left(A_\tau^{(T)}|\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, P, T\right) = \mathbf{M}^T_{(T(a_c),T(a_n))}R_{\left(T(a_c),T(a_f)\right)}A_{\left(T(a_c),T(a_f)\right)}. \tag{5.11}$$

While the repulsiveness of obstacles increases gradually with decreasing distance between the agent and the obstacle, observing attractive features like paths can instantly attract an agent to go towards it and then follow it.

Inspired by [124], the repulsiveness is modelled using a decay function considering the distance between the agent's current position and the position of the observed obstacle. To model the uncertainty in LP's decision, noise is added to the computed value. Similarly, the attractiveness of topography type path is modelled using the topography transition matrix (2.16). As a result, the attractiveness of the feature depends on the agent's encoded behaviour. If we consider a_p, a_w, and a_o to be the cells with first instance of path, water and obstacle along the agent's movement ray λ_τ; and define a_e to be the observable cell at the end of l_τ along λ_τ. a_f

is determined to be

$$a_f = \begin{cases} a_p & \text{if} \quad a_p \text{ exists else} \\ a_o & \text{if} \quad a_o \text{ exists else} \\ a_w & \text{if} \quad a_w \text{ exists} \\ a_e & \text{if} \quad \text{Otherwise} \end{cases}. \tag{5.12}$$

Considering the observability of obstacles, they can generally be categorised into two groups: *High Obstacles* (H) that can been seen from far like forests and *Low Obstacles* (L) that can only be seen from close proximity such as sudden drop in elevation or a river. Therefore,

$$R_{\left(T(a_c),T(a_f)\right)} = \begin{cases} R^H_{\left(T(a_c),T(a_f)\right)} & \text{if} \quad T(a_f) = \text{H and } d\left(a_f, a_c\right) < d^H \\ R^L_{\left(T(a_c),T(a_f)\right)} & \text{if} \quad T(a_f) = \text{L and } d\left(a_f, a_c\right) < d^L \\ 1 & \text{Otherwise} \end{cases}, \tag{5.13}$$

where $R^H_{\left(T(a_c),T(a_f)\right)}$ and $R^L_{\left(T(a_c),T(a_f)\right)}$ are the functions modelling the repulsiveness of the two obstacle types given that distance between the agent and the type of obstacle computed by $d(.,.)$ is less than d^H and d^L, the critical ranges for H and L respectively. The functions are given by

$$R^H_{\left(T(a_c),T(a_f)\right)} = a^H \exp\left(\frac{1 - \left(d^H - d\left(a_f, a_c\right)\right)}{b^H}\right) + n^r,$$

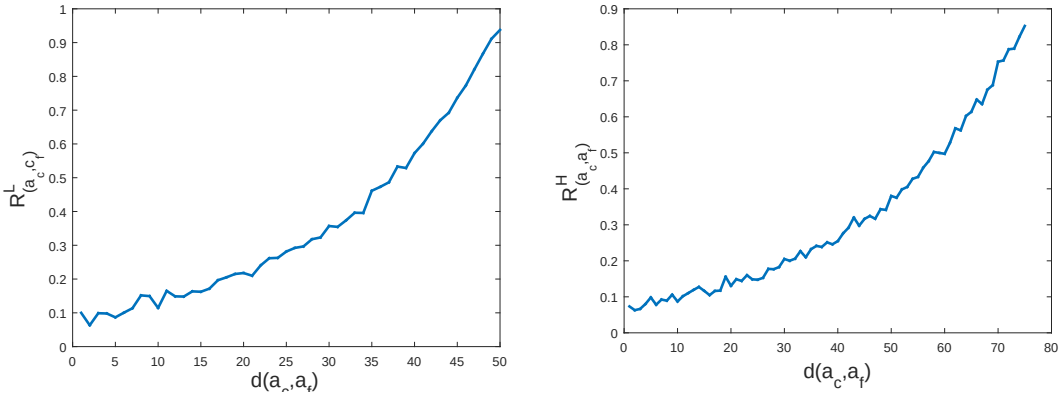
$$R^L_{\left(T(a_c),T(a_f)\right)} = a^L \exp\left(\frac{1 - \left(d^L - d\left(a_f, a_c\right)\right)}{b^L}\right) + n^r,$$

where a^H and a^L are the interaction strengths, b^H and b^L are ranges of repulsive interaction for H and L respectively and n^r is zero mean noise. Plot of the two functions are given in Figure 5.7.

In contrast, from a distance, only natural paths are considered attractive. Therefore

$$A_{\left(T(a_c),T(a_f)\right)} = \begin{cases} \mathbf{M}^T_{(T(a_c),Path)} & \text{if } T(a_f) = \text{Path} \\ \mathbf{M}^T_{(T(a_c),Ground)} & \text{Otherwise} \end{cases}, \tag{5.14}$$

The transition values $\mathbf{M}^T_{(T(a_c),Path)}$ and $\mathbf{M}^T_{(T(a_c),Ground)}$ are acquired from the topography transition matrix (2.16) depending on the topography feature type in a_c. Equations (5.13) and (5.14), allow agents to emulate a realistic path following and obstacle avoidance behaviour.



(a) Repulsiveness with respect to low obstacles **(b)** Repulsiveness with respect to high obstacles

Figure 5.7: Repulsiveness model for low and high obstacles respectively. The result includes random noise added.

Table 5.4: Parameter values for acceptance computation of global agent model.

Parameter	σ_λ	a^H	a^L	b^H	b^L	d^H	d^L	n^r
Value	$0.05°$	0.8	0.9	30m	20m	75 m	50 m	$G(0, 0.01)$

For example with obstacle avoidance, when an agent is faced with an obstacle along its heading, it would veer away from it. With tall obstacles, this happens at greater distance compared with low obstacles, which requires the agent to get close to the obstacle to notice it.

We show examples of these behaviours in the next section. Table 5.4 lists values of the new parameters defined for (5.13). Having presented the global agent model, we need to evaluate it, ensuring that with each improvement to the model, its behaviour improves too, producing realistic trails.

5.3 Global Agent Model Behaviour Evaluation

5.3.1 Environment Setup

The experiments are performed using both synthetic data sets and Sand Dunes data sets, the later presented in Section 2.5.2, illustrated in Figure 2.7.

5.3.2 Agent Setup

We use the same agent model setup detailed in Section 4.7.2 with few changes:

- The number of agent particles N is 50.
- The agent perception is modelled using the model detailed in Section 5.2.1
- The acceptance probability is computed using (4.10), (5.9), (5.10) and (5.11), modelling effects of search area on LPs local movement and global orientation.

- The agent bearing in (4.6) is modelled using (5.7).

The values for parameters used to model agent's behaviour, vision, strategy transition and move acceptance are given in Tables 2.3, 5.1, 5.2, 5.3 and 5.4 respectively. N^{FOV} and W^{FOV} configuration are modelled using 9 and 11 bins respectively. Unless stated otherwise for specific experiment, these remain the same for all experiments from here on.

5.3.3 Evaluation Results

5.3.3.1 Use of Long Term Memory and Direction Travelling Strategy

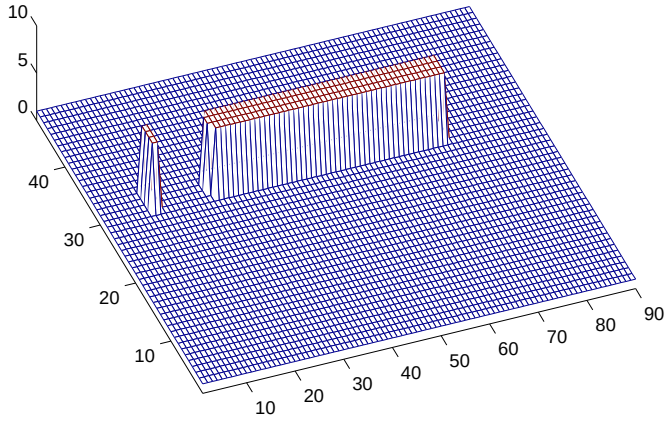
In this experiment, we use a synthetic setup where agents go to a place of interest if they observe one. They can decide to change places of interest when new ones are observed. Results of the experiment are shown in Figure 5.8. Figure 5.8b shows the case where agents do not forget features held in M_L. As a result most agents navigate around obstacles presented here by water and a wall to get to the intended place of interest (A) first observed when agents were initiated. However as the agents turn around the wall, they can see a new places of interest (B), as a result some agents change their intended destination to (B) and decide to go towards it. In both cases, the agent uses (5.9) to penalise proposed next positions that deviate from the direction of place of interest. Thus, they move around obstacles and get to place of interest, which at times can be out of view.

In contrast Figure 5.8c shows the case where agents do forget features not observed for a long time. As a result, agents forget place of interest (A) after a while (depends on ls_{max}) because they are not able to see it from the side of wall they are walking along. As a result, when some of the agents see place of interest (C), they decide to go towards this new observed place of interest. This shows that different distributions can emerge by allowing agents to forget or not to forget features in M_L.

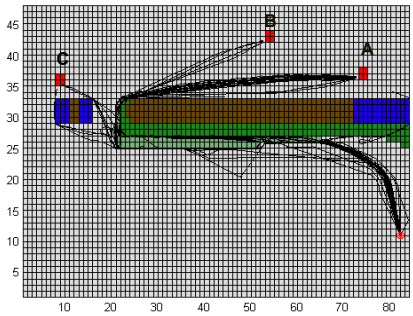
5.3.3.2 Agent Goal Dependent Strategies

Results of this experiment is illustrated in Figure 5.9.

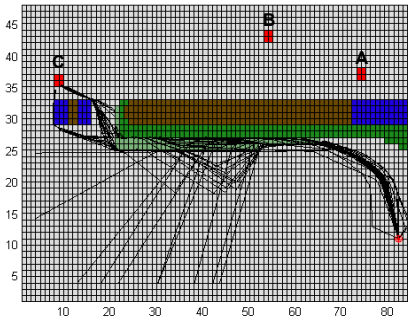
In this experiment, the probability of going to an observed place of interest is set to a high value ($p(DT) = 0.8$). As can be seen agents change strategies depending on the their state and the environment feature classes observable to them. Most agents prefer to travel along the natural paths, however when natural paths are not observable, agents resort to random travelling. In cases where a place of interest is seen, agents perform direction travelling and go towards the place of interest. The interesting behaviour that can be observed is when agents perform direction travelling, and are close to paths, they follow the path to the point where the path digresses from the place of interest. Some agents decide to stay put and become stationary.



(a) Elevation model



(b) Without forgetting behaviour



(c) With forgetting behaviour

Figure 5.8: Scenarios showing not forgetting and forgetting features in M_L using 50 agent trail samples. Agents initiate from position corresponding to the red encircled star. The red square regions labelled A, B and C represent the three places of interest. The blue cells represent water, grey cells a high wall green cells vegetation. The black lines represent agent trails.

5.3.3.3 Importance of Computing Global Orientation with Respect to Topography and Elevation

This relates to computing global orientation with respect to elevation, topography and obstacle avoidance using (5.10), (5.11) and (5.13) respectively. We illustrate this using the synthetic scenario in Figure 5.10, where the effects of obstacles (high and low) on the movement of the agents are illustrated. There are several points that can be observed:

- *When global orientation is not considered:*
  - *When* $p(DT) = 0.05$ *:* Agents do not go towards places of interest. Because agents do not have full knowledge of the environment and their movement are a function of their local vision, agents get very close to obstacle and are unable to explore the

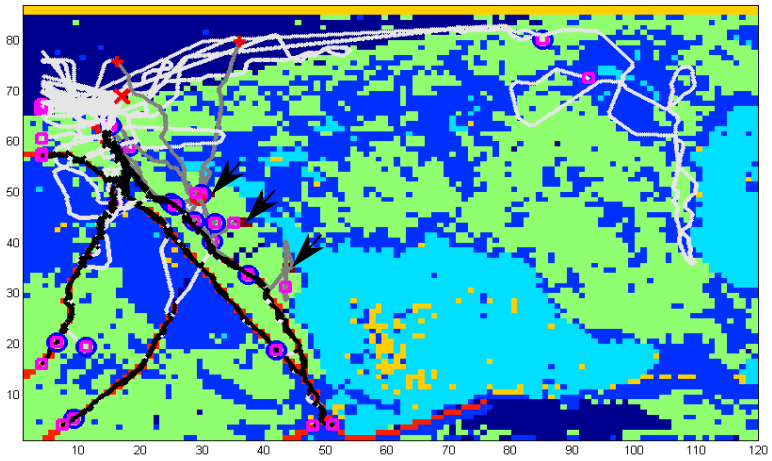


Figure 5.9: Various strategies performed at Sand Dunes.The agent trails are overlaid over the classified image of the area. White Line represent random travelling, black line represent path travelling, red X represent the PLS, magenta square represent the end location, blue circles represent stay put, red pluses represent the point place of interest was observed and Gray line represent direction travel. The arrow indicates the location of the place of interest target goal P_Int.

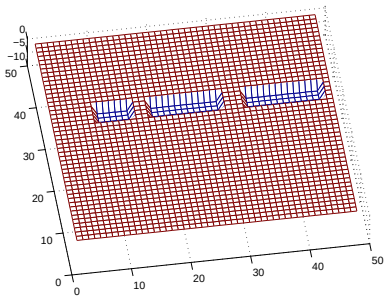
gaps that connect regions of the environment. This is illustrated in Figure 5.10c.

– *When* $p(DT) = 0.7$*:* Only a few agents can see places of interest on the opposite side of obstacles and decide to go towards them as illustrated in Figure 5.10d. Some agents initially decide to go towards a place of interest and then change their decision, thus the straight trails through the gaps.

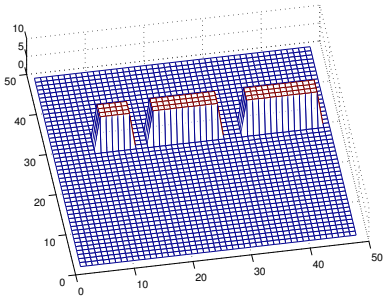
- *When global orientation is considered:*

– *When* $p(DT) = 0.05$*:* Agents do not go towards places of interest. They traverse the environment avoiding obstacles. With tall obstacles (Hs), agents decide to veer away earlier compared with low obstacles (Ls). This is illustrated in Figure 5.10e and Figure 5.10f respectively.

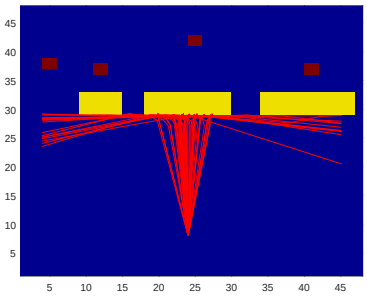
– *When* $p(DT) = 0.7$*:* In comparison to Hs, with Ls, because agents have greater visual access to the environment, more agents can see places of interest. As a result, more agents decide to go towards them. This difference can be seen in Figures 5.10g and 5.10h respectively.



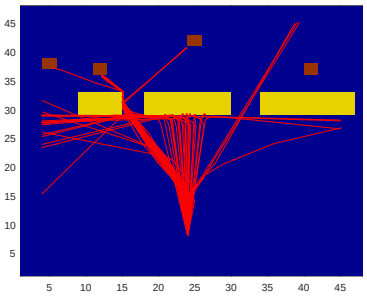
(a) Low obstacle (L) elevation



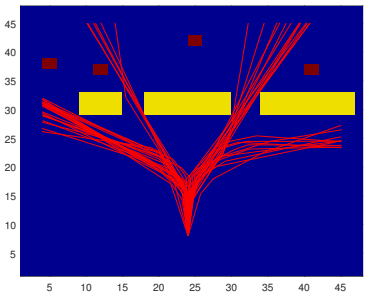
(b) High obstacle (H) elevation



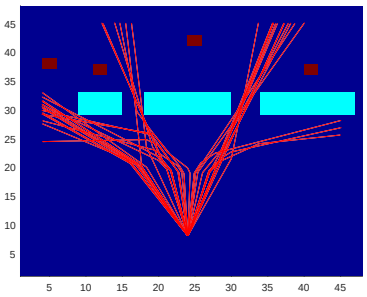
(c) H obstacle, no global orientation, do not consider place of interest



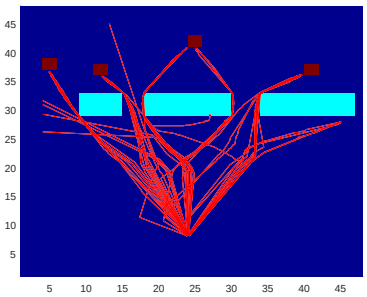
(d) H obstacle, no global orientation, consider place of interest



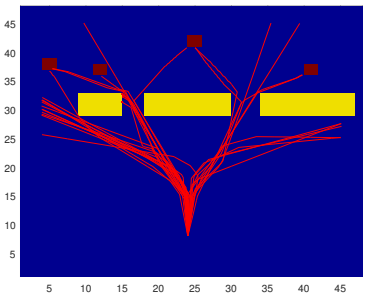
(e) H obstacle, global orientation, do not consider place of interest



(f) L obstacle, global orientation, do not consider place of interest



(g) L obstacle, global orientation, consider place of interest



(h) H obstacle, global orientation, consider place of interest

Figure 5.10: How modelling the agent's global orientation with respect to the topography and elevation of the environment; and place of interest affect it's movement and way finding capability. Figures on the top show the elevation data sets of the scenarios. In Figures 5.10c, 5.10e and 5.10f, the agents are configured so that they do not want to go towards places of interest (I). In Figures 5.10d, 5.10h and 5.10g on the other hand the agents are configured so that they can decide to go towards any of the places of interest observed. Yellow, red and light blue coloured cells represent High Obstacles (H), places of interest and Low Obstacles (L) (water stream) respectively.

5.4 Speed and Energy Consumption

Although the agent model presented in Section 5.2 is a good representation of how an LP would move in wilderness, it still does not consider a couple of important state depended behaviours: speed variation as a result of interaction with non-uniform surface of wilderness and energy consumption. In this section, we will incorporate both of these into the agent's state and kinematics model and show how it affects agent movement.

5.4.1 Agent State and Kinematics Model

To model energy consumption, we add a new term e, the energy level, to the state

$$\mathbf{t}_k = \left\{ \begin{bmatrix} x & y & \lambda & g & s & e & d \end{bmatrix}^T \; M_{S1} \; M_{S2} \; M_L \right\}_\tau . \tag{5.15}$$

At the start of simulation, the state is initialised to

$$\mathbf{t}_L^{(i)} = \left\{ \begin{bmatrix} x & y & \lambda & g & s & e_{max} & 0 \end{bmatrix}^T \; M_{S1} \; M_{S2} \; M_L \right\}_L ,$$

To incorporate the speed variations and energy consumption, we compute s_τ and e_τ using

$$\begin{aligned} s_\tau &= \mathrm{f}\left(\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, s_{max}(P), E, V, T\right) \\ e_\tau &= e_{\tau-1} - \mathrm{f}(\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, E, V) \end{aligned} . \tag{5.16}$$

Each agent starts with a maximum energy level $e_{max}(P)$, which is decremented as the agent traverses the environment, and when $e_\tau < e_{min}$, the agent stops. Thus, with p defined to be a variable specifying when stay put condition should be checked, the agent trail termination check is done by Algorithm (7).

5.4.2 Rate of Energy Consumption

Each agent is initiated with maximum energy to spend $e_{max}(P)$ given as $J\ Kg^{-1}\ m^{-1}$, where J stands for Joules (unit of work or energy expended), Kg stands for kilogram weight of the person and m is meters of altitude change. The amount of energy assigned depends on the LPs profile P i.e. his physical built and health. The relationship between the metabolic energy cost of walking and slope is modelled by a 5^{th} order polynomial regression [134]

$$\begin{aligned} \mathrm{f}(\tilde{\mathbf{t}}_\tau^{(i)}, \mathbf{t}_{\tau-1}, E, V) = 280.5 \times S^5 - 58.7 \times S^4 - 76.8 \times S^3 \\ + 59.9 \times S^2 + 19.6 \times S + 2.5J\ Kg^{-1}\ m^{-1} \times n, \end{aligned} \tag{5.17}$$

Algorithm 7 Agent trail termination check

TerminateTrajectory()
INPUTS: τ
OUTPUTS: run
if $\tau \% p == 0$ **then**
  if $rnd > p(SP)$ **then**
    run = false;
  end if
end if
if $run == true$ AND g_τ is reached **then**
  run = false;
else if $d_\tau > d_{max} || \tau > \tau_{max}$ **then**
  run = false;
else if $e_\tau < e_{min}$ **then**
  run = false;
end if

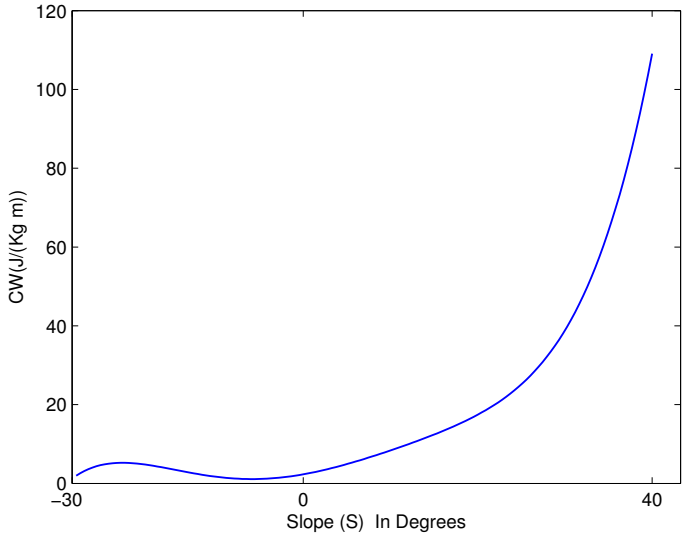


Figure 5.11: The relation between metabolic energy cost of walking and slope for hikers [134]. The minimum of walking cost is at slopes just below zero ($-5°$). This because people tend to reduce speed when going downhill. But due to the active control required to control the descent, cost of walking increase at slopes lower then $-5°$.

where S is the slope between agent's state at times $\tau - 1$ and τ, and n models additional factors that influence the rate of energy consumption such as vegetation density. Considering $e = 1$, the relationship between the metabolic energy cost of walking and slope is illustrated in Figure 5.11.

In normal circumstances, agents gradually looses energy with minimum energy cost. When walking uphill cost of walking increases as a function of the slope. When walking downhill, it is at its lowest at slopes of $-5°$ and then it increases as the slope becomes increasingly negative.

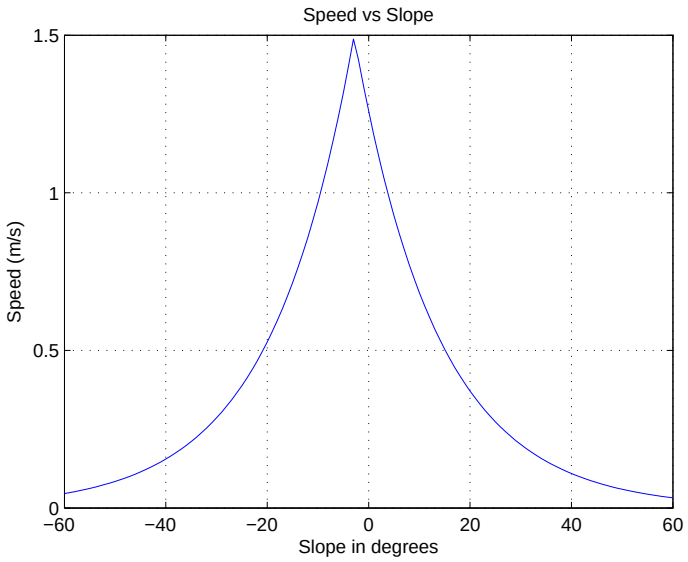


Figure 5.12: The relation between speed and slope for hikers [135]. The minimum speed happens at slope just below $0°$. But as soon as the slope downhill increases to steeper angles, the speed increases. This correlates with cost of walking with respect to slope given in [134].

Let $\tilde{\mathbf{t}}_\tau^{(i)} \in a_i$ to be the cell the agent is proposing to move to

$$n = \begin{cases} 5 & \text{if} \quad V(a_i) = \text{Dense vegetation} \\ 1 & \text{if} \quad V(a_i) = \text{Sparse vegetation} \\ 3 & \text{if} \quad V(a_i) = \text{Medium vegetation} \end{cases} . \tag{5.18}$$

5.4.3 Speed Model

Although the speed of the LP is bounded in the range $0 \leq s_\tau \leq s_{max}(P)$, where $s_{max}(P)$ depends on the LP's profile, the agent's current speed is influenced by the features in the environment. We model this using Tobeler's hiking function [135] as

$$s_\tau = s_{max}(P) \exp\{(-3.5 \times (|S| + 0.05))\}, \tag{5.19}$$

where $|S|$ is the absolute value of the slope between $\mathbf{t}_{\tau-1}$ and $\tilde{\mathbf{t}}_\tau^{(i)}$ giving the speed vs slope plot shown in Figure 5.12. As can bee seen in the plot, the speed decreases both with increasing and decreasing slope, with the maximum walking speed happenings at slope of around $-3°$. This reflects the fact that humans try to reduce their speed at lower slopes so that they can have more control over their body and speed. The speed of an LP is further affected by the vegetation and topography of the environment [135]. Therefore,

$$s_\tau = s_{max}(P)\exp\{(-3.5 \times (|S| + 0.05))\}s_\tau^{(V)}s_\tau^{(T)}, \tag{5.20}$$

where $s_\tau^{(V)}$ and $s_\tau^{(T)}$ models the effects of vegetation and topography respectively. Considering that $\tilde{\mathbf{t}}_\tau^{(i)} \in a_i$, $s_\tau^{(V)}$ and $s_\tau^{(T)}$ are given by

$$s_\tau^{(T)} = \begin{cases} 0 & \text{if } T_{(a_i)} = \text{Obstacle} \\ 0.1 & \text{if } T_{(a_i)} = \text{Water} \\ 1 & \text{otherwise} \end{cases}. \tag{5.21}$$

$$s_\tau^{(V)} = \begin{cases} 0.3 & \text{if } V_{(a_i)} = \text{Dense vegetation} \\ 1 & \text{if } V_{(a_i)} = \text{Sparse vegetation} \\ 0.6 & \text{if } V_{(a_i)} = \text{Medium vegetation} \end{cases}. \tag{5.22}$$

The effect of terrain topography and elevation on both agent's speed and energy consumption is illustrated in Figure 5.13. The plots show two paths by an agent. One goes down a valley, the other climbs a hill. As can be seen the energy consumed is higher for the agent which climbs up the valley side. In both cases, speed reflect the elevation changes. When going uphill/downhill, speed reduces. The flat surfaces in speed variation (plot 5.13e and plot 5.13j) are due to constant speed when traversing through a cell. The plots also show the maximum speed each agent has.

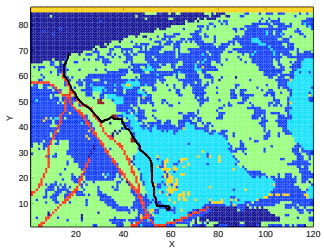
The effect of different initial maximum energy settings is illustrated in Figure 5.14. As can be seen with high initial energy (typically for a person that is physically build and young), the agents are able to traverse greater distances compare to agents initialised with lower energy (i.e. people who are not well, old or disabled.). In the later case, the agents gets fatigued quite quickly and stop moving.

5.5 Global Agent Model Evaluation

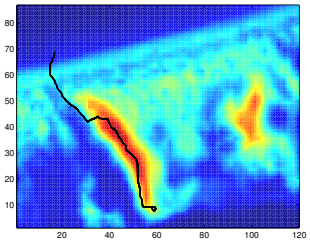
In this section we evaluate the proposed agent model and its ability to produce faithful representation of the LP trail. We perform two experiments:

- First experiment: In this experiment we first generate two distribution using the agent and the diffusion models. We then assess the result (the generated distributions) with respect to the distribution of observed GPS logs collected (Section 4.6), shown in Figure 4.10.

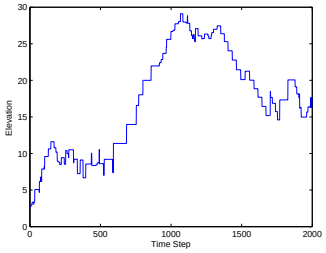
- Second experiment: In the second experiment, we evaluate the effects of having a more informed initial distribution in the task of search for a LP. We perform a simulation study



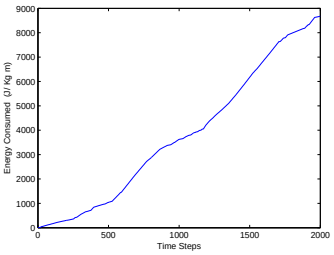
(a) Sample 1 over the classified land cover



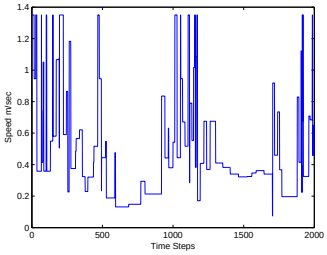
(b) Sample 1 over elevation.



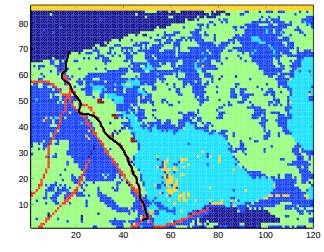
(c) Sample 1 elevation changes



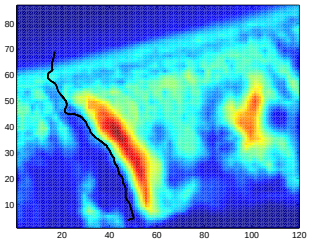
(d) Sample 1 energy consumption



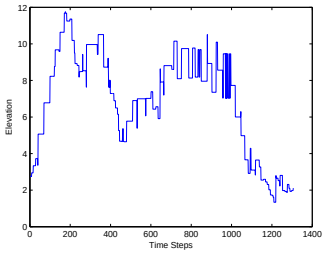
(e) Sample 1 speed variation



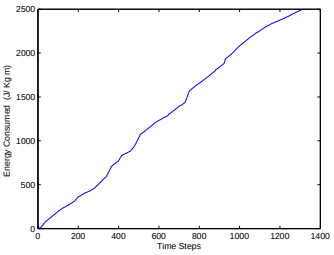
(f) Sample 2 over the classified land cover



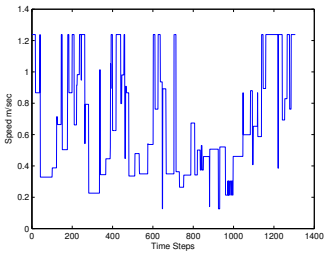
(g) Sample 2 over elevation



(h) Sample 2 elevation changes

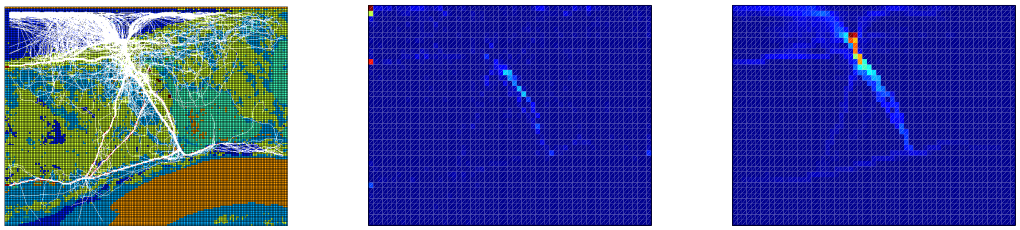


(i) Sample 2 energy consumption

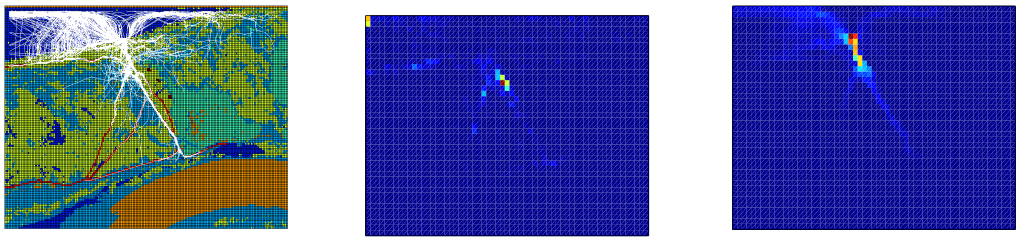


(j) Sample 2 speed variation

Figure 5.13: Speed variation and Energy consumption of an agent while moving uphill and downhill. The Figures in row 1 and row 3 show the agent trail sample overlaid over the classified topography and elevation model of the environment and the elevation of cells traversed. Plots in row 2 and 4 show the energy consumption and speed variations.



(a) Agent trails **(b)** Distribution over LP location **(c)** Distribution over LP Trail



(d) Agent trails **(e)** Distribution over LP location **(f)** Distribution over LP Trail

Figure 5.14: Agent-based distribution with different energy settings. The Figures on the top show generated agent-based distribution, with agents initialised with $21KJ/Kgm$. The Figures on the bottom show generated agent-based distribution, with agents initialised with $3KJ/Kgm$.

and compare the result of search for a lost target using both initial distributions typically used in search and rescue operations (detailed in Section 2.4), and initial distribution generated using the agent model.

5.5.1 Environment Setup

Both experiments are performed using New Forest data sets shown in Figure 4.8. For the second experiment, looking at Figure 5.15, we treat the base location used for data collection experiment as the PLS and four points selected at random on GPS logs of participants as points where LP became static i.e. their end locations.

To generate the initial distribution, while for agent model, we use the same search area data set discretisation of $(5$m on a side for both experiments, for diffusion model, we use two different discretisations.

5.5.1.1 First Experiment

The size of the search area is 595m $\times$ 845m, and is discretised into equal cells of $(5$m on a side, with similar discretisation of the search data sets E, T and V. This results in a 119×169 gridded representation.

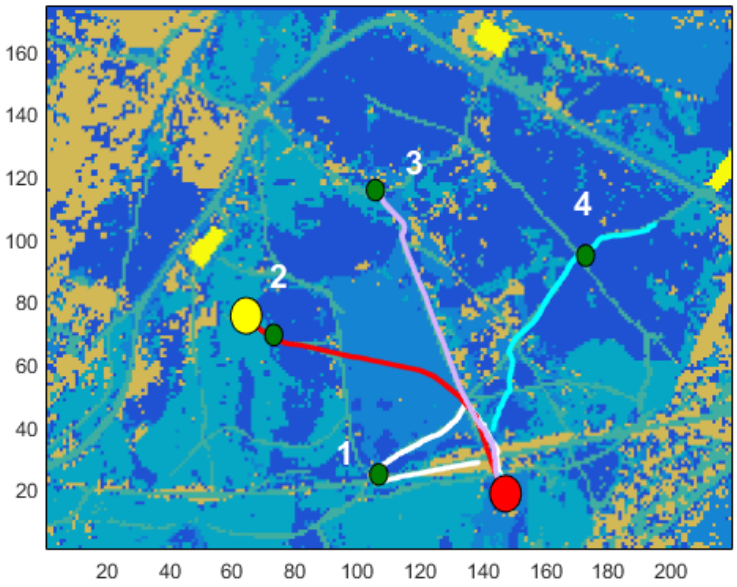


Figure 5.15: Experimental scenario for global agent model evaluation. Four GPS track logs in New Forest are shown. The green circles represent the points along the trails that have been selected to serve as end points or locations where the LP is considered to have stopped. The red and yellow circles represent the PLS and end target locations.

5.5.1.2 Second Experiment

For the second experiment, we consider bigger search area $800\text{m} \times 1100\text{m}$, discretised into equal cells of 10m on a side.

5.5.2 Unmanned Aerial Vehicle Path Planning

For the search process in the second experiment, the UAV is assumed to be point-like with a perfectly known pose and a static camera frame of $(10\text{m})^2$, which is reflected in representation of results. To navigate the search area, the UAV paths needs to be computed. We use the look-ahead path planning method detailed in Section 2.3.6.

The discrete Bayesian formulation presented in Section 2.3.4 is used to perform the update. The observation model described in Section 2.3.5 is used to investigate the search area for the presence of the LP with *missed detection* and *false detection* rates specified in Section 2.6.

5.5.3 Lost Person Movement Model

5.5.3.1 Agent Model Setup

The agent model follows the setup detailed in Section 5.3.2, where $N = 1000$ particles and agent state is given by (5.15). The agent speed and energy consumption is computed using (5.16). The agent trail is terminated when termination criteria is met given in algorithm (7),

The agent model is run for 1560 time steps, equivalent to 13 walking time, and d_{max} of

1170m.

5.5.3.2 Diffusion Model Setup

Details of the diffusion model setup is given in Section 2.5.3.1. For the first experiment, the length of each time-step is $\Delta\tau = 3.3s$ – the average amount of time required for a person to walk from the centre of one cell to the centre of an adjacent cell of 5m resolution. This means a simulation run of 237 simulation steps with step length of 5m would model 13 minute walking time and 1170m walking distance. For the second experiment, $\Delta\tau = 6.6s$ and the simulation is run for 117 steps.

5.5.4 Evaluation Metrics

5.5.4.1 First Experiment

For the first experiment, we need to use a metric that can give us an indication of the dissimilarity between distributions. Both the agent model and the GPS tracks are continuous-valued. However, the diffusion model can only specify probability over a grid. Therefore, we transform the agent-based distribution and GPS measurements into distributions onto a grid for direct comparison. Several dissimilarity measures exist for comparing the different distributions.

- Kullback Leibler Divergence (KLD): Suppose that we have two distributions P and Q that we want to compare, the KLD [136] is defined to be

$$d_{KL}(P||Q) = \sum_i P_i \log \frac{P_i}{Q_i}, \tag{5.23}$$

  where i is index of a cell in the discretised representation of the distributions. From the information theory point of view, the KLD has the property that it measures how inefficient on average it would be to code one distribution using the other.

- χ^2 Statistic: This distance measures is given by:

$$d_{\chi^2}(P,Q) = \sum_i \frac{(P_i - m_i)^2}{m_i}. \tag{5.24}$$

  It is used to measure how unlikely it is that one categorical or bind distribution is drawn from the population represented by the other categorical bind distribution.

- Minskowski-Form Distance: This distance measures is given by

$$d_{L_r}(P,Q) = \left(\sum_i |P_i - Q_i| \right)^{1/r}. \tag{5.25}$$

This measure with L_1 is often used to compute dissimilarity between colour images [137].

The disadvantage of these distance measures are that they are applied for *bin-by-bin* dissimilarity measures and are sensitive to bin size [138]. Using these dissimilarity measures, only pairs of bins that have the same index in the two distributions are compared and matched. In our case, if the agent model predicted the LP location one cell to the left of where it is, we would get a huge error. Therefore, we need a dissimilarity that considers the correspondence between bins in the two distributions. One such dissimilarity measure is Earth Movers Distance (EMD) [139].

Earth Mover's Distance EMD: . First introduced in computer vision community [140], EMD measures dissimilarity of two bind distributions considering the correspondence between bins and ground distance (the amount of distance between bins). Intuitively, given two distributions, one can be considered as a mass of earth properly spread in space, the other as collection of holes in that same space. Then the EMD measures the least amount of work needed to fill the holes with earth. Here a unit of work corresponds to transporting a unit of earth by a unit of ground distance.

For example, given two distributions P and Q each having n bins, a flow matrix $\mathbf{F}$, where $f_{i,j}$ indicates flow (i.e. earth) to move from P_i to Q_j and a ground distance matrix $\mathbf{D}$ where $d_{i,j}$ models the ground distance between i^{th} bin to j^{th} bin, EMD is given by

$$EMD(P,Q) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} f_{i,j}d_{i,j}}{\sum_{i=1}^{n}\sum_{j=1}^{n} f_{i,j}}. \tag{5.26}$$

Intuitively, $d_{i,i} = 0$ when $i = j$ and gets bigger with larger $d_{i,j}$ – the distance between i and j in the bin space.

Computing EMD is based on a solution to the transportation problem [141]. For simplicity, we assume our distributions P and Q are collection of one-dimensional arrays of "bins" for which the EMD can be efficiently computed by scanning the arrays and keeping track of how much earth needs to be transported between consecutive bins in an array. For example assuming both P and Q are of the same size, EMD is given by Algorithm 8.

Algorithm 8 EMD computation

$\mathbf{GridBasedSearch}()$
for $i = 0$: Number of rows **do**
  $EMD[i, 0] = 0$
  for $j = 1$: Number of columns $- 1$ **do**
    $EMD[i, j] = (P_{i,j} + EMD[i, j - 1]) - Q_{i,j}$
  end for
end for
$EMD(P, Q) = \sum_i \sum_j |EMD|$

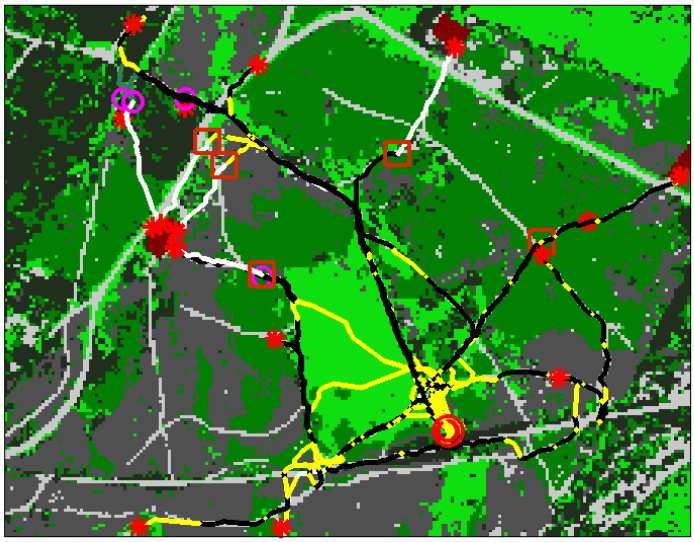


Figure 5.16: Several agent trail samples. The colours of the background image denote pixel-wise classification. The strategies used in each trail are colour coded. Route travelling, random travelling, direction travelling and stay put strategies are represented by black, yellow, white and red stars respectively. The red circle and square represents PLS and the location from which a candidate target was observed.

For the first experiment, in order to show the convergence of the two models, we start by running both models for a set time (13 minutes) assessing the EMD distance measure for each of the generated distributions with respect to the GPS log distribution. The distribution with lower EMD measure represents higher similarity than distribution with higher EMD measure.

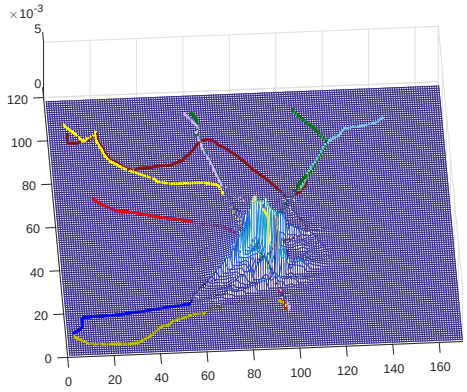
5.5.4.2 Second Experiment

To compare search result using the different prior distributions, we use the ”time-to-locate” metric, which is the number of cells visited and revisited before the LP is detected. This decision is made when $\max_{a_i \in \mathcal{A}} p(\mathbf{t}_S \in a_i) >= \gamma$, where $\gamma = 0.8$.

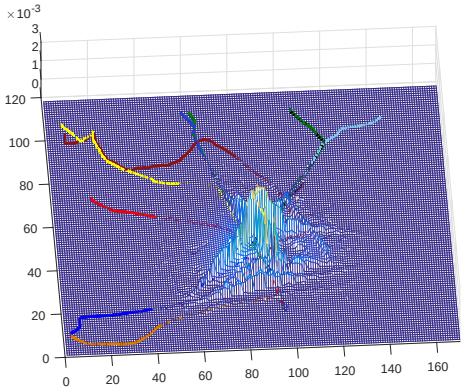
5.5.5 Evaluation Results

5.5.5.1 First Experiment

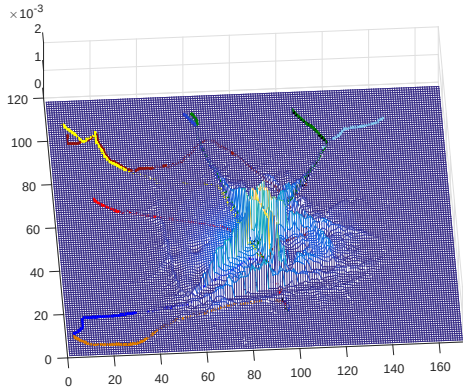
A number of simulated agent trails are shown in Figure 5.16. These illustrate the effect of goal switching. Goal switching occurs as a result of the agent observing different features in the environment. For example, all agents follow linear features and paths of least resistance reflecting the encoded behaviour. However, when they see a car park, some of them decided to switch to the Direction Travelling goal (shown in white) and head directly towards it considering it to be the target destination. This is similar to the behaviour seen from the actual logs shown in Figure 4.10a.



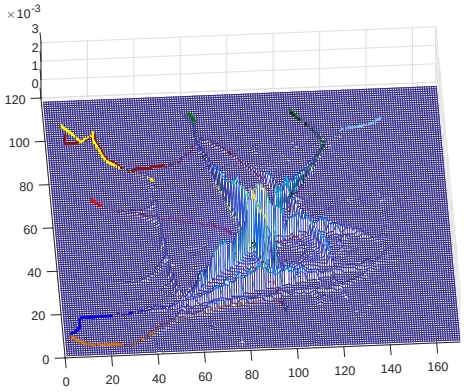
(a) Diffusion model at 13 mins.



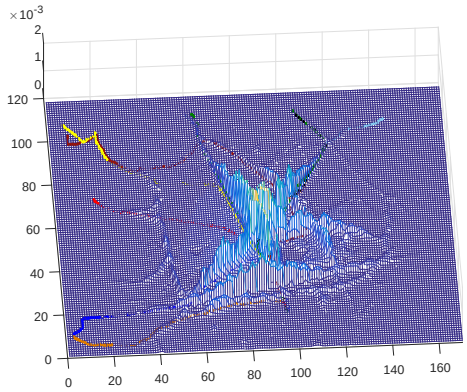
(b) Diffusion model at 24 mins.



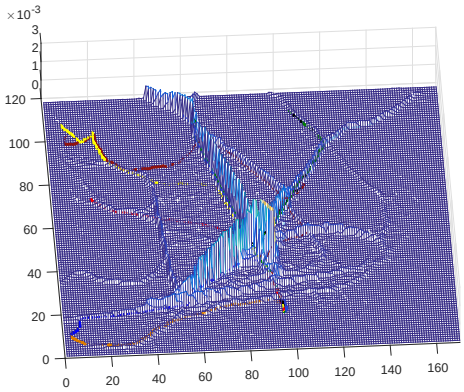
(c) Diffusion model at 37 mins



(d) Diffusion model 48 mins



(e) Diffusion model 58 mins



(f) Agent model 13 minutes

Figure 5.17: The computed initial distributions using the diffusion-based human movement model and the proposed agent-based human movement model. The higher peaks represents higher probability of LP being in that area. The observed trails are given with a unique colours each.

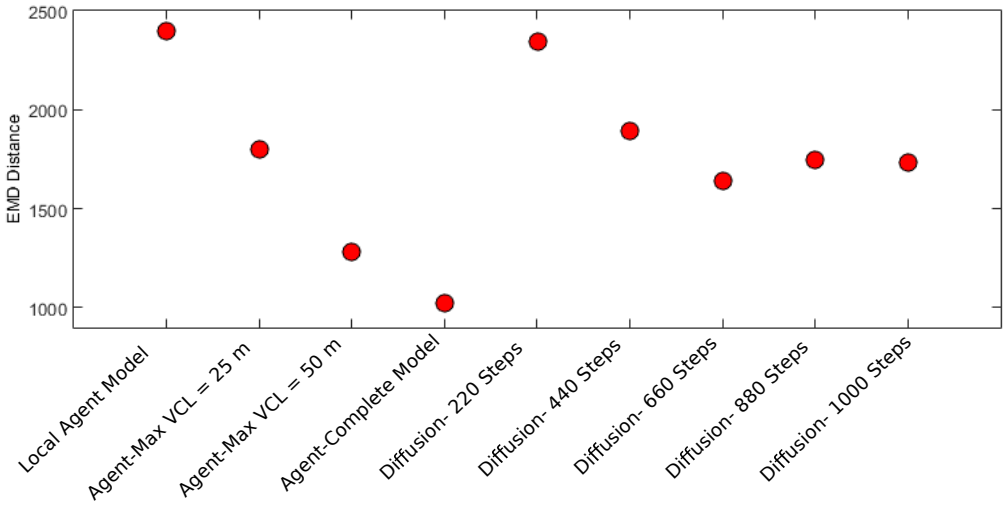
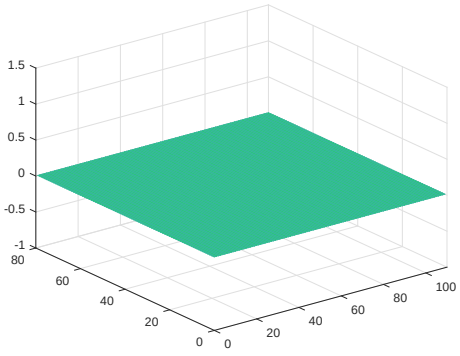


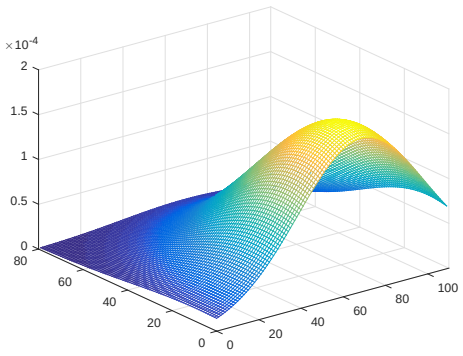
Figure 5.18: EMD measure for both agent and diffusion-based initial distributions with respect to GPS logs. The EMD measure is given with respect to different agent model configurations and different time durations for diffusion model.

Figure 5.17 shows the initial distributions generated using both the agent model and the diffusion model with the latter for different time durations. Considering the average speed of participants, and running both models for simulation time equivalent to the duration of experiment (13 minutes), the distribution generated using the agent model was able to faithfully reflect the distribution of GPS logs in contrast to diffusion model generated distribution. This conclusion is supported by the EMD measures of both distributions with respect to the distribution of observed trails (GPS logs) shown in Figure 5.18. The diffusion model performance was only improved by running the simulation for simulation time equivalent to 58 minutes illustrated in Figure 5.17e, and confirmed by improved EMD measures. The results suggest that the agent model tends to take more directed, focused routes which accurately model the directed behaviour of the LPs. The diffusion model however, is less focused and its uncertainty gradually fills the map. This suggests that the diffusion models both take longer to develop, and are likely to generate less efficient distributions for the search algorithms. Furthermore, the diffusion algorithms can only take account of local features. They cannot model distant features, nor can they model the impact of an agent switching between strategies.

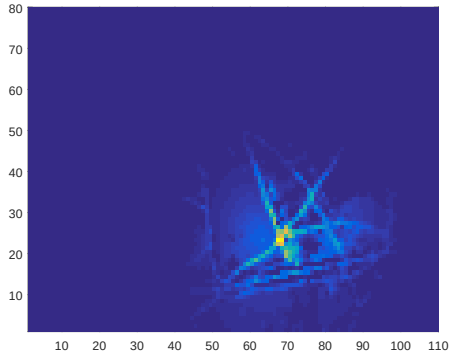
To examine the effect of spatial dependency of features. We performed two experiments: first, where only the agent short vision length $S_{VCL} = 50$m is considered and second, where the length of S_{VCL} is reduced to 25m. We can see from the EMD measures that this has a significant impact on the generated distributions. As the VCL is reduced, the EMD increases, showing that the computed trail becomes less and less accurate.



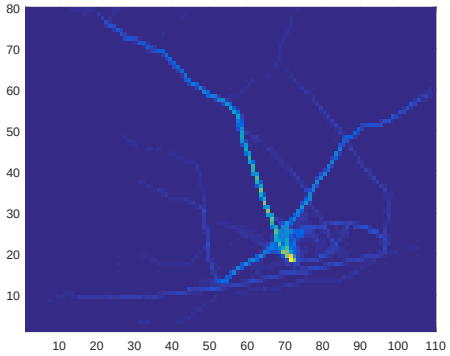
(a) Uniform initial distribution



(b) Gaussian-based initial distribution



(c) Diffusion-based initial distribution



(d) Agent-based initial distribution

Figure 5.19: Different types of initial distribution for a search scenario in New-forest.

The final test we undertook was using the local agent model detailed in Section 4.4. This as expected, resulted in high EMD measures similar to diffusion model for time duration of 13 minutes. The reason it is not exactly the same is due to the differences in their modelling of the LP movement.

These results show that there is a greater advantage in using an agent models over a diffusion model to capture the LP's movements.

5.5.5.2 Second Experiment

The different initial distribution types used to perform the search are illustrated in Figure 5.19. For Gaussian distribution, $\mu = \begin{bmatrix} 70 & 10 \end{bmatrix}$ and $\sigma = \begin{bmatrix} 30\text{m} & 0 \\ 0 & 30\text{m} \end{bmatrix}$. Results of the experiment is given in Table 5.5.

From the results we can see that search with agent-based initial distribution results in least time-to-locate metric. The results show that by using the agent-based initial distribution, the system was able to perform the search task significantly faster than using any of other initial distribution types. This is confirmed by the UAV trajectory plot for target 1 using the different

Table 5.5: The ”time-to-locate” factor with respect to four different type of initial prior distributions for the four different target end locations (based on actual GPS logs of people).

Trial	1	2	3	4
Uniform prior	1189	3950	4388	4262
Gaussian-based prior	2465	3633	3355	2383
Diffusion-based prior	1700	3177	2213	3018
Agent-based prior	1196	3222	223	515

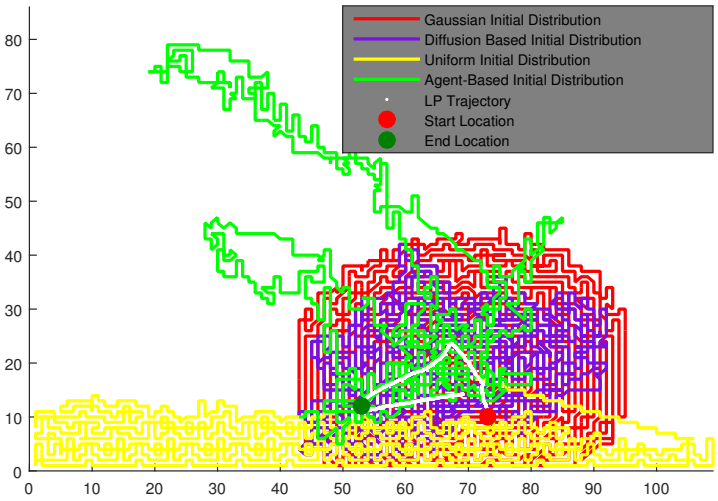


Figure 5.20: The UAV search path using the different types of initial distribution.

initial distribution types in Figure 5.20.

As can be seen, with uniform initial distribution, a kind of sweep search is performed until the LP is located. This takes very long. The search is improved by using Gaussian-based initial distribution, where the UAV follows the contours of the distribution until the LP is located. However, the UAV still has to search many areas that are not traversable according to the search area data sets, hence waste of search time and effort. Using the diffusion-based initial distribution, search is further improved, but still the UAV search areas that are least likely to have been traversed by the LP. This is remedied using agent trail-based initial distribution. With the agent trail-based initial distribution, the UAV is able to search for the LP along the likely paths the LP may have taken (over traversable regions of the search area), thus reducing both search time and effort. This results in much shorter search times. From the results, we can conclude that the more refined and accurate the initial distribution is, the less time it takes for the UAV to locate the LP.

5.6 Summary

In this chapter, we presented the global agent model of the LP movement. The novelty of the agent model is that we explicitly model lost behaviour: rather than progressing with a fixed goal to a fixed destination, the agent has no set destination and can switch between goals. We model the LP's perception, interaction with the environment, and local and global decision making. In addition, we model and argue that the LP's speed and energy consumption plays a vital role in determining how far they can travel.

Through experiments, comparing the agent model with the diffusion model, we showed that not only the agent generated distribution is a more faithful representation of the LP trail, it also helps significantly reduce the search times.

Our implementation was prototyped in MATLAB, and so direct performance comparisons in terms of speed of computations are not meaningful. However, there are many efficient libraries for parallel operations such as MASON [142], which could be used.

The main limitation of the agent model is that it consists of many parameters. To get optimal performance out of the agent model, the next challenge is to develop a systematic way to tune the parameters values.

Chapter 6

Agent Model Calibration

6.1 Introduction

The previous two chapters introduced the agent model, capturing LP's behaviours such as local and global interaction with the environment. To model these behaviours, the model uses a large number of parameters whose values must be tuned. This process is known as agent model calibration, and is the focus of this chapter.

We start by reviewing existing approaches in Section 6.2. Because of the number of parameters and relationships between them in our proposed agent model, it is very hard to calibrate them directly. Therefore, we first perform parameter screening using a Design Of Experiment (DOE) approach to identify the sensitive parameters that have the greatest influence on the behaviour of the agent in Section 6.3. Then, having identified the sensitive parameters, we calibrate them using observed data, using the process detailed in Section 6.4. After calibrating the model, in Section 6.5 we describe the verification and validation of the model. Verification and validation are the processes of showing respectively, the correctness of model construction and the truthfulness of a model with respect to its problem domain i.e. modelling lost people movement. Finally, the the chapter is summarised in Section 6.6.

Key Terms

- **Calibration:** This refers to tuning parameter values using observed data.
- **Sensitivity Analysis:** This refers to study of agent model parameters and determining the set of parameters that have most effect on model outputs.
- **Prior :** Also called prior distribution refers to the initial distribution specified for agent model parameters.
- **Posterior:** Also called posterior distribution refers to calibrated value/distribution for a parameter.

6.2 Parameter Calibration Approaches

Calibration is the process of setting the values of parameters to improve model agreement with an observable reality [143].

Suppose $\mathbf{\Theta}$ is the set of parameters used by the agent model. To calibrate their values, a set of measurements $\mathbf{Z}_{log}$ are collected. Using these, the maximum a posterior estimate of the parameters is computed from

$$\mathbf{\Theta}^* = \arg\max_{\mathbf{\Theta}} p(\mathbf{Z}_{log}|\mathbf{\Theta})p(\mathbf{\Theta}). \tag{6.1}$$

Although the importance of calibration is widely understood in the ABM community [77, 92, 98, 99, 144], there are very few published papers in which it has been applied [111]. In the published literature, depending on the availability of observed data, calibration is generally performed either using expert judgement or observed data.

In cases where observed data is hard to gather for models like the ones in [131,145], model parameters are calibrated simply by manually adjusting the parameter values and checking that simulation results are plausible. In case of [131], it is checked if the simulation results meets a pre-determined results and [145] checks if it meets the expert opinion of what should happen in certain scenarios considering plausibility of various actions. Both these methods are subject to imperfect human perception of how the agent should move.

When it is possible to collect observed data, again, calibration is done either by manual adjustment of parameters or by using a calibration method. For example in [111, 146], video recording of pedestrians are used for manual calibration of agent model parameters. It is done by extracting parameter specific information using manual annotation of pedestrians in the video and following their movement and interaction with environment and other pedestrians.

Although manual tuning of parameters can work with these models as they only have a few parameters, it can quickly become inefficient, time consuming and can lead to sub-optimal results, specifically when the model in question consists of many parameters.

As a result, some researchers prefer to use calibration methods for the task [95, 147, 148]. For example the work in [95], introduces a two step process. In the first step, observed data gathered through image processing methods from video of an intersection crossing is used to collect the pedestrian's speed and acceleration data. In the second step, a search algorithm is used for calibration, which constantly changes the model parameter values until the root mean square difference between the simulated and real attributes are minimised

Although these calibration methods automate the process of calibration, they make certain

assumptions:

- It is assumed that agent model dynamics/transition can be modelled using only few parameters.
- All parameters have similar level of impact upon the results. In reality some parameters can have significant effect on the result and others not as much.

Our proposed agent model contains many parameters that control its mechanics such as vision, memory, kinematics, and behaviours. Assuming that the parameters used for the mechanics are optimally defined through vigorous tests during the model development, we are interested in calibrating only the parameters used to encode the agent's behaviours (the transition matrices (2.15), (2.16) and (2.17)). However, we are still left with 30 parameters (including agent's speed s_{max}) to tune. Defining this set of parameters to be $\boldsymbol{\Theta} = \{S_1, \cdots, S_4, V_{1,1}, \cdots, V_{3,3}, T_{1,1}, \cdots, T_{4,4}, s_{max}\}$[1] , to find the value of $\boldsymbol{\Theta}^*$, we marginalise over agent trails,

$$\boldsymbol{\Theta}^* = \arg\max_{\boldsymbol{\Theta}} p(\mathbf{Z}_{log}|\boldsymbol{\Theta})p(\boldsymbol{\Theta}) = \arg\max_{\boldsymbol{\Theta}} \int p(\mathbf{Z}_{log}|\mathbf{T}_B)P(\mathbf{T}_B|\boldsymbol{\Theta})p(\boldsymbol{\Theta})d\mathbf{T}_B, \tag{6.2}$$

However, this optimisation problem is extremely challenging because of the high-dimensional nature of the parameter space and the non-linear nature of the model.

To address the problem of dimensionality, we use parameter screening techniques to identify the parameters which have the greatest impact on the computed trails. The non-linearity in the model is addressed through the use of Monte Carlo methods [149, 150].

6.3 Parameter Screening

Parameter screening is done through performing a sensitivity analysis, which considers the robustness of model results with respect to relatively small perturbations in the parameters [143].

To identify parameters for the calibration process, we decompose $\boldsymbol{\Theta}$ into two disjoint sets, $\boldsymbol{\Theta} = \boldsymbol{\Theta}_0 \cup \boldsymbol{\Theta}_1$ where $\boldsymbol{\Theta}_0$ are the sensitive parameters, and $\boldsymbol{\Theta}_1$ are all other non-sensitive parameters. Given this decomposition, the calibration process can concentrate on $\boldsymbol{\Theta}_0$; nominal values can be used for $\boldsymbol{\Theta}_1$. In other world, we seek

$$\begin{aligned} \boldsymbol{\Theta}_0^* &= \arg\max_{\boldsymbol{\Theta}_0} p(\mathbf{Z}_{log}|\boldsymbol{\Theta}_0; \boldsymbol{\Theta}_1)p(\boldsymbol{\Theta}_0)p(\boldsymbol{\Theta}_1) \\ &= \arg\max_{\boldsymbol{\Theta}_0} \int p(\mathbf{Z}_{log}|\mathbf{T}_B)P(\mathbf{T}_B|\boldsymbol{\Theta}_0; \boldsymbol{\Theta}_1)p(\boldsymbol{\Theta})d\mathbf{T}_B. \end{aligned} \tag{6.3}$$

[1] We have not included S_5, sines we know that it is physically extremely difficult for hikers to traverse the slope range in this class.

Table 6.1: Example of Design Of Experiment for system with 3 parameters.

Design Points	x_1	x_2	x_3	Response Measures
$d^{(1)}$	-1	-1	-1	$y^{(1)}$
$d^{(2)}$	+1	-1	-1	$y^{(2)}$
$d^{(3)}$	-1	+1	-1	$y^{(3)}$
$d^{(4)}$	+1	+1	-1	$y^{(4)}$
$d^{(5)}$	-1	-1	+1	$y^{(5)}$
$d^{(6)}$	+1	-1	+1	$y^{(6)}$
$d^{(7)}$	-1	+1	+1	$y^{(7)}$
$d^{(8)}$	+1	+1	+1	$y^{(8)}$

If the optimisation is to be successful, it must be the case that $p(\mathbf{T}_B|\mathbf{\Theta}_0^*;\mathbf{\Theta}_1)\approx p(\mathbf{T}_B|\mathbf{\Theta}^*)$. We use the DOE approach to perform sensitivity analysis and determine if a given parameter lies in $\mathbf{\Theta}_0$ or $\mathbf{\Theta}_1$.

6.3.1 Design of Experiment Approach

An experiment is an orderly procedure carried out with the goal of verifying, refuting, or establishing the validity of a hypothesis [151]. Experiments vary greatly in their goal and scale, but always rely on a repeatable procedure and logical analysis of the results. Experiments provide insight into cause-and-effect by demonstrating what outcome (*response*) occurs when a particular process variable (*factor*) is manipulated by setting *factor level* signifying range of variable (factor) values [152]. A combination of levels for all factors is called a *design point*.

The (statistical) Design Of Experiment (DOE) is an efficient procedure for planning experiments so that the response can be analysed to yield valid objective conclusions [153]. It is used for laying out of the detailed experimental plan in advance of doing the experiment. Well chosen experimental designs maximises the amount of information that can be obtained for a given amount of experiment effort.

For example, defining x and y as factor and response variables respectively, in a linear model with three factors x_1, x_2 and x_3, the DOE is given in Table 6.1, where $d^{(i)}$ is the i^{th} design point or the i^{th} row of factor level settings for x_1, x_2 and x_3, and $y^{(i)}$ is the response variable for $d^{(i)}$. We use +1 and -1 for the factor level setting, also called *coding the data*. The table formed by the columns x_1, x_2 and x_3 is called the design Table or Design Matrix.

DOE begins with determining the objectives of an experiment and selecting several discrete or continuous independent input factors that can be controlled and one or more measured dependent output responses. Experimental data are used to derive an empirical model linking the outputs and inputs. These empirical models generally contain first and second order terms

Table 6.2: Full factorial and complete interaction DOE for system with 3 parameters with one-way, two-way and three-way interaction.

Design Points	x_1	x_2	x_3	x_1x_2	x_2x_3	x_1x_3	$x_1x_2x_3$	Response Measures
$d^{(1)}$	-1	-1	-1	+1	+1	+1	-1	$y^{(1)}$
$d^{(2)}$	+1	-1	-1	-1	+1	-1	+1	$y^{(2)}$
$d^{(3)}$	-1	+1	-1	-1	-1	+1	+1	$y^{(3)}$
$d^{(4)}$	+1	+1	-1	+1	-1	-1	-1	$y^{(4)}$
$d^{(5)}$	-1	-1	+1	+1	-1	-1	+1	$y^{(5)}$
$d^{(6)}$	+1	-1	+1	-1	-1	+1	-1	$y^{(6)}$
$d^{(7)}$	-1	+1	+1	-1	+1	-1	-1	$y^{(7)}$
$d^{(8)}$	+1	+1	+1	+1	+1	+1	+1	$y^{(8)}$

and are called *meta-models*. The most common empirical models take a linear form, which is

$$y^{(i)} = \beta_0 + \beta_1 x_1^{(i)} + \beta_2 x_2^{(i)} + \beta_{12} x_1^{(i)} x_2^{(i)} + e^{(i)}, \tag{6.4}$$

where $x_1^{(i)}$ and $x_2^{(i)}$ are values for factors 1 and 2 in the i^{th} design point, β_0 is the intercept, β_1 and β_2 are main effect of factors $x_1^{(i)}$ and $x_2^{(i)}$ respectively, β_{12} is the interaction effect of $x_1^{(i)}$ and $x_2^{(i)}$; and $e^{(i)}$ is the approximation error. The constant β_0 is the response of $y^{(i)}$ when the main effects and the interaction effect are 0.

When the experimental data are analysed, all unknown parameters $\beta_0, \beta_1, \beta_2, \beta_{12}$ are estimated and tested to see which ones are significantly different from 0.

There are many DOE methodologies, each used to achieve a different objectives. Since our goal is to decompose agent model parameters $\mathbf{\Theta}$ and identify sensitive parameters, we perform DOE with the objective of screening the parameters. A design used to achieve this is factorial DOE [154–158], which is an experimental setup that consists of multiple factors and their separate and conjoined influence on the dependent variable or response.

6.3.1.1 Factorial DOE

There are two types of Factorial DOEs: *Full factorial design* and *fractional factorial design*. Full factorial designs are used to model main effects and interaction of parameters. For example, for a system with three factors ($q = 3$) and two factor levels setting ($m = 2$), the full factorial design consists of the $N = m^q = 8$ design points illustrated in Table 6.2. As can be seen, using the full factorial design captures all the main effects (x_1, x_2, x_3), two-way interactions (x_1x_2, x_2x_3, x_1x_3) and three-way interactions ($x_1x_2x_3$). The key requirement of full factorial design is that they need to be orthogonal, which means:

- The sum of each column is zero.

$$\sum_i x_j^{(i)} = 0 \ \ \forall j,$$

  where j stands for j^{th} variable.

- The sum of the products of the elements of any two columns is zero.

$$\sum_i x_j^{(i)} x_l^{(i)} = 0 \ \ \forall j \neq l.$$

Orthogonality guarantees that we can always estimate the effect of one factor or interaction clear of any influence from any other factor or interaction.

The main disadvantage of full factorial DOE is that they are only efficient for systems with small number of factors. The number of design points grow exponentially with an increasing number of system parameters. For example for a model with $q = 30$ and only two factor levels, this model would require $N = 2^{30} = 1,073,741,824$ design points. To deal with this problem, fractional factorial DOE designs have been proposed [159].

6.3.2 Fractional Factorial Design

Fractional factorial designs are experimental designs that use a subset of the design points from factorial designs. The main requirement is that they need to be orthogonal same as full factorial designs.

Fractional Factorial Design consists of $N = m^{q-p}$ design points where p describes the size of the fraction of the full factorial used. For example, for a system with 5 parameters ($q = 5$) with two factor levels ($m = 2$) for each parameters, we want to reduce the factorial design to a quarter of the full factorial design ($p = 2$). As a result we get $N = 2^{5-2} = 8$ design points.

To compute the fractional factorial design matrix, we use *design generators* and consider *confounding* or *aliasing* effects of parameters. The later is when two factor effects cannot be separately estimated and are determined from design generators. We show the process of generating a fractional factorial design using an example. Consider a scenario where a system has 5 factors ($q = 5$), and the DOE uses $m = 2$ factor level setting. But there is only time to simulate 8 design points. To achieve this, we need to design a fractional factorial design matrix with $N = 2^{5-2} = 8$ design points.

First we compute the full factorial design for $q - p$ parameters, which in this case is 3 parameters. This would give us a design matrix given in Table 6.2.

In this full factorial design, there tend to be redundancies in terms of number of interactions

Table 6.3: Full factorial and complete interaction DOE for system with 5 parameters, one and two-way interaction .

Design Points	x_1	x_2	x_3	x_1x_2	x_2x_3	x_4	x_5	Response Measures
$d^{(1)}$	-1	-1	-1	+1	+1	+1	-1	$y^{(1)}$
$d^{(2)}$	+1	-1	-1	-1	+1	-1	+1	$y^{(2)}$
$d^{(3)}$	-1	+1	-1	-1	-1	+1	+1	$y^{(3)}$
$d^{(4)}$	+1	+1	-1	+1	-1	-1	-1	$y^{(4)}$
$d^{(5)}$	-1	-1	+1	+1	-1	-1	+1	$y^{(5)}$
$d^{(6)}$	+1	-1	+1	-1	-1	+1	-1	$y^{(6)}$
$d^{(7)}$	-1	+1	+1	-1	+1	-1	-1	$y^{(7)}$
$d^{(8)}$	+1	+1	+1	+1	+1	+1	+1	$y^{(8)}$

Table 6.4: Fractional factorial DOE for system with 5 parameters and no interaction terms.

Design Points	x_1	x_2	x_3	x_4	x_5	Response Measures
$d^{(1)}$	-1	-1	-1	+1	-1	$y^{(1)}$
$d^{(2)}$	+1	-1	-1	-1	+1	$y^{(2)}$
$d^{(3)}$	-1	+1	-1	+1	+1	$y^{(3)}$
$d^{(4)}$	+1	+1	-1	-1	-1	$y^{(4)}$
$d^{(5)}$	-1	-1	+1	-1	+1	$y^{(5)}$
$d^{(6)}$	+1	-1	+1	+1	-1	$y^{(6)}$
$d^{(7)}$	-1	+1	+1	-1	-1	$y^{(7)}$
$d^{(8)}$	+1	+1	+1	+1	+1	$y^{(8)}$

that can be estimated i.e. columns x_1x_2 or x_2x_3 or $x_1x_2x_3$. Specifically, column that models highest order of interaction are negligible. Therefore we start with these, we substitute these to represent factor level setting for remaining factor i.e.

$$x_5 = x_1x_2x_3. \tag{6.5}$$

Since there is only one three-way interaction and that is set to model x_5 factor level setting, any one of the remaining two two-way interaction can be used to model x_4. In this case, we decided it to be

$$x_4 = x_1x_3. \tag{6.6}$$

This results in the design matrix in Table 6.3.

Removing the remaining interaction columns, we are left with the matrix in Table 6.4. Now to determine the *design generator* for (6.5), we multiply both side by x_5, we get $x_5x_5 = x_1x_2x_3x_5$. Since $x_5x_5 = I$ a unity (vector with all ones),

$$x_5x_5 = x_1x_2x_3x_5 \rightarrow I = x_1x_2x_3x_5. \tag{6.7}$$

Similarly, for (6.6), we get

$$x_4 x_4 = x_1 x_3 x_4 \rightarrow I = x_1 x_3 x_4. \tag{6.8}$$

The terms on the right of (6.7) and (6.8) are design generators. Using these design generators, we can generate different configurations of design matrix. The total collection of generators for the design, including all new generators that can be formed as product of these generators is called *defining relation* for the design. In this case, the algebraic multiplication of (6.5) and (6.6) results in

$$x_5 x_4 = x_1^2 x_2 x_3^2 \rightarrow x_5 x_4 = I x_2 I \rightarrow x_5 x_4 = x_2.$$

After some algebraic manipulation, this results in third design generator $I = x_2 x_4 x_5$. Thus the defining relation for the design in Table 6.4 is

$$I = x_1 x_2 x_3 x_5 = x_1 x_3 x_4 = x_2 x_4 x_5. \tag{6.9}$$

In general, there will be $m^p - 1$ generators in a defining relations for 2^{q-p} fractional factorial design. Now, we can use this defining relation to identify the design resolution and confounding parameters and parameter interactions.

The length of the shortest generator in the defining relation is called the *Resolution* of the design. A design resolution determines the complexity of the meta-model, computational cost of the DOE and the degree to which estimated main effects are aliased (or confounded) with estimated two-way, three-way or higher interactions.

High resolution designs allows complex measurements to be made, requiring exponentially increasing computational times [153, 154, 159] and mainly used to determine multiple factor (parameter) interaction effects with no confounding factor effects. In the case of example in Table 6.4, because the shortest design generator is of length 3 (6.9), the design resolution is three (III).

Defining the design resolution to be R, a design can model interaction of order less than $R - 1$. Thus, with the design resolution of example above, it is able to model interactions of order less then $3 - 1 = 2$.

To show this, we identify confounding relations for one of the factors in DOE in Table 6.4. We multiply it to the defining relation (6.9). For example, to identify confounding relationships

of term x_1, we get

$$x_1 I = x_1 x_2 x_3 x_5 x_1 = x_1 x_3 x_4 x_1 = x_2 x_4 x_5 x_1 \rightarrow x_1 = x_2 x_3 x_5 = x_3 x_4 = x_1 x_2 x_4 x_5 \quad (6.10)$$

This means that the main effect of term x_1 is confounded or aliased with two-way interaction $x_3 x_4$, three-way interaction $x_2 x_3 x_5$ and four-way interaction $x_1 x_2 x_4 x_5$. The same process can be used to identify confounding relationships of each main term and two-way and three-way interactions.

Because we are interested in determining the set of parameters that have significant impact on the generated distribution $p(\mathbf{T}_B|\Lambda)$, we consider only the main effects. Therefore, we use the *resolution three* (III) fractional factorial design.

6.3.3 Agent Model Sensitivity Analysis

To perform sensitivity analysis of 30 parameters in $\mathbf{\Theta}$ with 2 factor level settings, we use a resolution three fractional factorial design with $2^{(30-26)}$ design points resulting in the DOE design matrix with 16 rows and 30 columns[2] illustrated in Appendix E.1. To account for variability of the response, each $d^{(i)}$ was replicated 2 times, which is typical in the agent model sensitivity analysis [158].

In our model, all factors are quantitative. Since each parameter in $\mathbf{\Theta}$ is initially specified in terms of a mean and standard deviation, factors represent the mean of distributions keeping the standard deviation of the parameters constant throughout the experiment. Each factor has two levels (low and high), which are set based on there plausibility using

$$x_j^{(i)} = \mu_j^{(i)} + (\mu_j^{(i)} \times c_j \times d_j^{(i)}), \quad (6.11)$$

where j stands for j^{th} variable, i stands for i^{th} experiment or design point, c_j is the perturbation magnitude of the parameter and $d_j^{(i)}$ is factor level encoded by -1 and $+1$.

For each design point, after computing the value of each factor using (6.11), three processes take place. First, in a post selection operation, each row in each mean transition matrix is normalised.

Second, a distribution over the LP trail is generated using N particles, each represented by the agent model of the LP detailed in Chapter 5 with sampled behaviour in the form of transition matrices (2.15), (2.16) and (2.17) from distributions represented by the computed normalised mean values and the original standard deviations specified in Table 2.3.

[2]The design matrix was generated using MATLAB functions fracfactgen() and fracfact() respectively.

Third, a cost function is calculated for the generated distribution with respect to a baseline distribution. In our case, the baseline distribution is generated using factor level value 0, i.e. no perturbation to the model parameter mean values. This gives the response variables y for each design point.

6.3.3.1 Cost Function

Defining the baseline distribution over LP trajectory to be $p(\mathbf{T}_B^b|\Lambda)$, generated with factor level setting of 0 for all factors, and $p(\mathbf{T}_B^i|\Lambda)$ to be the distribution generated with factor level setting in $d^{(i)}$. To calculate the cost function for response measure $y^{(i)}$, we use a distance measure between $p(\mathbf{T}_B^b|\Lambda)$ and $p(\mathbf{T}_B^i|\Lambda)$. The distance measure we use is Kullback Leibler Divergence (KLD), therefore $y^{(i)} = D_{KL}\left(p(\mathbf{T}_B^i|\Lambda)||p(\mathbf{T}_B^b|\Lambda)\right)$, where

$$D_{KL}\left(p(\mathbf{T}_B^i|\Lambda)||p(\mathbf{T}_B^b|\Lambda)\right) = \sum_{m\in\mathcal{A}} p(a_m \in \mathbf{T}_B^i|\Lambda)\log\left(\frac{p(a_m \in \mathbf{T}_B^i|\Lambda)}{p(a_m \in \mathbf{T}_B^b|\Lambda)}\right). \tag{6.12}$$

Recall that $p(a_m \in \mathbf{T}_B^i|\Lambda)$ is the probability that cell a_m lies on trail $\mathbf{T}_B^i$ and is computed from the indicator function $T(\mathbf{T}_B^j, a_i)$ defined in Section 3.3.2. To ensure that the KLD can be computed in the case where a_i does not appear in both distributions, we first set all such cells with a very small non-negative value and then normalise the distribution. Having computed the response measures for each design point, we need to analyse them.

6.3.3.2 Result Analysis Methods

We use two analysis methods: *linear regression meta-model* and *main effects plots*. In the linear regression meta-model, it is assumed that there is a linear dependency between the input and response variables. Therefore, the goal is to fit the linear regression model,

$$y^{(i)} = \beta_0 + \sum_{j=1}^{J} \beta_j x_j^{(i)} + e^{(i)}, \tag{6.13}$$

and use the main effects to determine the sensitive parameters. The main effects plot on the other hand visually illustrates the sensitivity of the model to changes in the value of factors. In a two level factor setting, this is done by plotting the average response when a factor is set to high (or +) value and average response response when the factor is set to low (or -) value. For example, in case of design matrix in Table 6.2, if we assume that $y^{(1)} = 8$, $y^{(2)} = 3$, $y^{(3)} = 5$, $y^{(4)} = 0$, $y^{(5)} = 6$, $y^{(6)} = 3$, $y^{(7)} = 2$, $y^{(8)} = 9$. Defining e_1 and e_2 to denote the the average

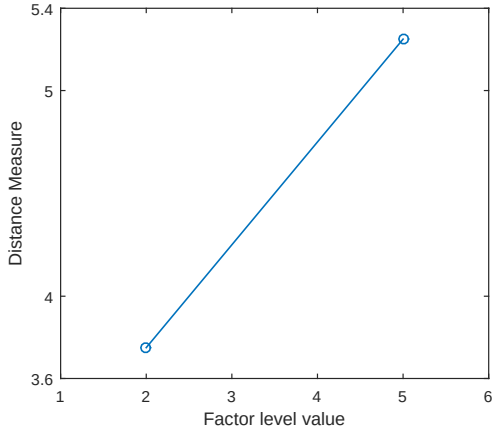


Figure 6.1: Example of the main effects plot.

response with factor x_1 set to high and low values respectively

$$e_1 = \frac{3+0+3+9}{4} = 3.75 \;\; ; \;\; e_2 = \frac{8+5+6+2}{4} = 5.25,$$

the main effects plot is given by plotting the e_1 and e_2 illustrated in Figure 6.1. Using this, the factors can be ranked from least to most sensitive.

6.3.3.3 Sensitivity Analysis Results

The full sensitivity analysis of the agent model is presented in Appendix E.2. Any parameter with $p\_value < \alpha$ (the significance level), where $\alpha = 0.05$, is considered sensitive and added to the set of parameters that need calibration. This because, small perturbations in the value of these parameters introduce a big effect on $p(\mathbf{T}_B|\Lambda)$ resulting in larger KLD, compared to parameters with $p\_value > 0.05$, which have very small to negligible effect on $p(\mathbf{T}_B|\Lambda)$ with small perturbation in their values.

From Table E.1, we can see that there are only 14 parameters that have p_value less than α. This result is also reflected in the main effects plots presented in Appendix E.2.1. As can be seen the severity of the sensitive parameters vary. Transition to path feature ($T_{(2,4)}$, $T_{(3,4)}$, $T_{(4,4)}$), speed s_{max} and slope s_1 are most sensitive of the parameter set. Reflecting the importance of these parameters in real scenarios.

As a result, $\mathbf{\Theta}_0$ is 14 dimensional with the parameters:

$$\mathbf{\Theta}_0 = \{S_1, V_{(1,1)}, V_{(1,2)}, V_{(1,3)}, T_{(1,1)}, T_{(1,2)}, T_{(2,3)}, T_{(3,1)}, T_{(3,3)}, T_{(3,4)}, T_{(4,2)}, T_{(4,3)}, T_{(4,4)}, s_{max}\}, \tag{6.14}$$

and $\mathbf{\Theta}_1$ includes the rest:

$$\mathbf{\Theta}_1 = \{S_2, S_3, S_4, S_5, V_{(2,1)}, V_{(2,2)}, V_{(2,3)}, V_{(3,1)}, V_{(3,2)}, V_{(3,3)}, T_{(1,3)}, T_{(1,4)}, T_{(2,1)}, T_{(2,2)}, T_{(2,4)}, T_{(3,2)}, T_{(4,1)}\}. \tag{6.15}$$

6.4 Empirical Calibration of Agent Models

Given that the set of sensitive parameters have now been identified, we can calibrate their values using observed data.

6.4.1 Parameter Estimation Using Monte Carlo Techniques

Static parameter inference on non-linear and non-Gaussian models is very hard and mostly do not have closed form solutions. For this reason it is necessary to resort to approximations, which can be achieved through Monte Carlo methods such as Markov Chain Monte Carlo (MCMC) and Sequential Monte Carlo (SMC) methods. These methods are used to sample from complicated non-linear, multi-dimensional distributions [160–164]. While the former is used to construct a Markov chain on the state space, whose steady state distribution is the posterior distribution of interest, the later is used mainly for approximation of any non-linear, multi-dimensional sequence of probability distributions that change over time detailed in Section 3.3.1.

The only work we are aware that has attempted to calibrate a complex model such as ours is Goodrich's calibration of parameters used in a diffusion model [61], which uses MCMC Metropolis-Hastings. In this work, synthetic observation of path taken by a scout in the form of GPS log is used to calibrate the parameters of the model. The MCMC method used for calibration is an iterative method, with each iteration representing some perturbation of parameter value, by the way of methods such as random walk. At the end of the each iteration, the perturbed parameter value is either accepted or rejected based on a likelihood ratio test of current and previous iteration results.

Although MCMC methods have been proved to converge to a stationary distribution [165], they suffers from two main drawbacks. First, they requires thousands of iterations, which for a model like ours can be nontrivial, especially for long simulation times [166]. Second, a bad choice of proposal distribution, which guides the exploration of the parameter space, can make the performance of the method unreliable [150].

SMC too has its own drawbacks. Because SMC depends on the initial set of particles throughout, and with large time index k, the SMC approximations deteriorate, resulting particle degeneracy detailed in Section 3.3.1.

Recently, Chopin et al. proposed the SMC^2 method in [167], which brings the best of both SMC and MCMC together by combining Particle Filter (PF) and Iterated Batch Importance Sampling (IBIS) [168]. The PF provides a flexible representation of arbitrary distributions. The IBIS is an SMC based static parameter estimation method. To diversify particles, SMC^2 utilises MCMC rejuvenation steps periodically during the simulation. In addition to this, it can

dynamically increase the number of particles to ensure *particle impoverishment* (reduction of the number of distinct samples within a collection) does not happen even when time index is large.

Since SMC^2 performs the MCMC rejuvenation periodically in contrast to pure MCMC, its computational complexity is significantly less than iterative MCMC methods.

6.4.2 Parameter Estimation Using (SMC^2) Technique

SMC^2 treats the problem of computing $p(\mathbf{\Theta}_0|\mathbf{Z}_{log};\mathbf{\Theta}_1)$ as a recursive exploration of the sequence of posterior distributions. The process of SMC^2 calibration for our agent model is as follows:

- Step 1: The calibration is initiated by sampling from the parameter state, which we will call *parameter particles*. At $\tau = 1$, the full set of parameters are sampled,

$$\{\mathbf{\Theta}^{(m)} \sim p(\mathbf{\Theta}), w_\tau^{(p,m)} = 1/N_p\}_{m=1}^{N_p},$$

  where $w_\tau^{(p,m)}$ is the weight of the parameter particle $\mathbf{\Theta}^{(m)}$ at time-step $\tau = 1$. The superscript p stands for parameter. Considering only one of the parameter particles $\mathbf{\Theta}^{(m)}$, a Particle Filter (PF) with N_t particles is used to model the distribution over the Lost Person trail using it. Each particle is initiated by

$$\{\mathbf{t}_\tau^{(i,m)} \sim p(\mathbf{t}_L|P), w_\tau^{(t,i,m)}\}_{i=1}^{N_t},$$

  where $w_\tau^{(t,i,m)}$ is the weight of the i^{th} trail particle and $w_\tau^{(t,i,m)} = 1/N_t$ at time $\tau = 1$. We call these *agent particles*.

- Step 2: Each agent particles $\mathbf{t}_\tau^{(i,m)}$ representing trail sample state is propagated forward in time using the particle approximation model (3.3) with the parameter sample $\mathbf{\Theta}^{(m)}$.

- Step 3: At every n^{th} time-step, the agent particle weights are updated by

$$w_\tau^{(t,i,m)} \propto p_{\mathbf{\Theta}^m}\left(\mathbf{z}_{log,\tau}|\mathbf{t}_\tau^{i,m}\right) = \mathcal{G}(\mathbf{t}_\tau^{(i,m)}; \mathbf{z}_{log,\tau}, \Sigma_{\mathbf{z}_{log,\tau}}),$$

  a GPS observation likelihood model where $\mathbf{z}_{log,\tau}$ is the GPS observation, and $\Sigma_{\mathbf{z}_{log,\tau}}$ is the variance on the GPS measure at time-step τ.

- Step 4: The parameter particle weights are update by

$$w_{\tau}^{(p,m)} = w_{\tau-1}^{(p,m)} \left(\frac{1}{N_t} \sum_{i=1}^{N_t} w_{\tau}^{(t,i,m)} \right). \tag{6.16}$$

- Step 5: The agent particles are re-sampled according to

$$\mathbf{t}_{\tau}^{(i,m)} \sim Mult_N \left(\left\{ w_{\tau}^{(t,i,m)} \right\}_{i=1}^{N_t} \right),$$

  where $Mult_N(\cdot)$ is a multi-nomial re-sampling method such as the one in [169].

Steps 2 to 5 continues until parameter particle degeneracy occurs, which is tested for at every n^{th} time-step by computing the effective sample size (detailed in Section 3.3.1),

$$N_{eff} = \frac{\left(\sum_{m=1}^{N_p} w_{\tau}^{(p,m)} \right)^2}{\sum_{m=1}^{N_p} \left(w_{\tau}^{(p,m)} \right)^2} < \eta \, N_p, \tag{6.17}$$

where $\eta \in (0, 1)$. If $N_{eff} < \eta \, N_p$, re-sampling is performed. This consists of two steps:

- Step 6: Parameter particles are re-sampled according to

$$\mathbf{\Theta}^{(m)} \sim Mult_N \left(\left\{ w_{\tau}^{(p,m)} \right\}_{m=1}^{N_p} \right).$$

  Each re-sampled parameter particle inheriting its parents agent particles.

- Step 7: MCMC Metropolis-Hastings kernel [149,170] is used to rejuvenate the parameter particles.

In step 7, parameter particles are rejuvenated by proposing new N_p parameter particles, generated either independently using

$$\{ \tilde{\mathbf{\Theta}}_0^{(m)} \sim N \left(\hat{\mu}, \hat{\Sigma} \right), \tilde{w}_{\tau}^{(p,m)} = 1/N_p \}_{m=1}^{N_p}, \tag{6.18}$$

or using a Gaussian random walk

$$\{ \tilde{\mathbf{\Theta}}_0^{(m)} \sim N \left(\mathbf{\Theta}_0^{(m)}, c\hat{\Sigma} \right), \tilde{w}_{\tau}^{(p,m)} = 1/N_p \}_{m=1}^{N_p},$$

where

$$\hat{\mu} = \frac{1}{\sum_{m=1}^{N_p} w_{\tau}^{(p,m)}} \sum_{m=1}^{N_p} w_{\tau}^{(p,m)} \mathbf{\Theta}_0^{(m)},$$

$$\hat{\sum} = \frac{1}{\sum_{m=1}^{N_p} w_\tau^{(p,m)}} \sum_{m=1}^{N_p} w_\tau^{(p,m)} \left(\mathbf{\Theta}_0^{(m)} - \hat{\mu}\right) \left(\mathbf{\Theta}_0^{(m)} - \hat{\mu}\right)^{\mathbf{T}}.$$

The term c is a tuning constant used to achieve optimal scaling of the Metropolis-Hastings algorithm [170]. Ideally the tuning parameter c should be selected so that it both allows for fast changes in $\mathbf{\Theta}_0$ and yields a high probability of acceptance. Unfortunately these are two competing goals. If we choose a proposal distribution with a small c, the probability of acceptance will be high, however the resulting Markov chain will be highly correlated as the $\mathbf{\Theta}_0$ change only very slowly. If on the other hand, we choose proposal distribution with a large c, the $\mathbf{\Theta}_0$ can potentially move very fast, however the probability of acceptance will be rather low. For this reason, finding a the ideal c is considered an art.

Since parameters in the transition matrices need to be normalised, step 7 results in proposed sampled value for the complete parameter set $\mathbf{\Theta}$. Using the proposed parameter samples, the following steps are performed:

- Step 8: New N_t agent particles are initiated and propagated forward $1 : \tau$ for each of the proposed parameter particle using Steps 1 through to 5, resulting in $\left\{\tilde{\mathbf{\Theta}}^{(m)}, \tilde{\mathbf{t}}_{1:\tau}^{(i,m)}, \tilde{w}_\tau^{(p,i)}\right\}_{i=1}^{N_t}$.

- Step 9: At time τ, the ratio r of proposed parameter particles are computed using

$$r = \tilde{w}_\tau^{(p,m)} \; / \; w_\tau^{(p,m)},$$

  where $w_\tau^{(p,m)}$ is accumulated total weight of the parameter particle m and $\tilde{w}_\tau^{(p,m)}$ is that of proposed parameter particle m computed using (6.16).

- Step 10: Each new proposed parameter particle is accepted with probability $\min(1,\ r)$ by first drawing a sample u from a uniform distribution $\mathcal{U}(0, 1)$ and then accepting the proposed particle

$$\left\{\mathbf{\Theta}^{(m)}, \mathbf{t}_{1:\tau}^{(i,m)}, w_\tau^{(p,i)}\right\}_{i=1}^{N_t} \leftarrow \left\{\tilde{\mathbf{\Theta}}^{(m)}, \tilde{\mathbf{t}}_{1:\tau}^{(i,m)}, \tilde{w}_\tau^{(p,i)}\right\}_{i=1}^{N_t},$$

  if $r \ < \ u$.

In the event of low acceptance rate, in Step 10, the number of agent particles N_t is increased to $N_t^* = N_t + n_t$ and the new agent particles are sampled for each parameter sample $\mathbf{\Theta}^{(m)}$. The new agent particles are propagated for time $1 : \tau$ once again using Steps 1 through to Step 10, setting $N_t = N_t^*$.

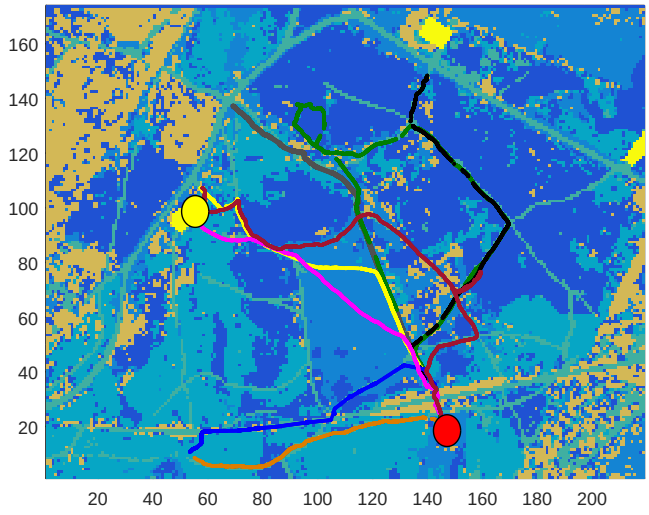


Figure 6.2: Global Positioning System logs used for calibration. The red circle represents the PLS and the yellow circle represents the target location.

Having done this, agent particles are again checked for degeneracy using (3.7). If $N_{eff} < \eta\ N_p$, steps 9 through to step 10 are repeated until $N_{eff} > \eta\ N_p$. In which, case Steps 1 to 5 are repeated.

6.4.3 Calibration Results

The calibration was performed for 14 parameters identified through sensitivity analysis in Section 6.3.3. From the collected GPS logs of data collection experiment detailed in Section 4.6, 8 logs (illustrated in Figure 6.2) are used to calibrate the agent model.

The calibration method was initialised with $N_p = 150$ parameter particles and $N_{\mathbf{t}} = 100$ agent particles. The number of agent particles was increased by 50 whenever the acceptance ratio of MCMC step went below 0.15. The upper bound for the number of agent particles were set to 400. The agent particle update and re-sampling took place at every 15^{th} time-steps.

The effective parameter sample size was computed using (6.17) with $\eta = 0.5$. If deemed degenerate, particle samples were re-sampled and then rejuvenated in step 7 with $c = 0.1$ to allow for small steps in the parameter space increasing the acceptance rate.

Because we are using a gridded environment, from implementation point of view, we first computed the GPS observation likelihood for each cell using the Algorithm in (9), which resulted in the likelihood map illustrated in Figure 6.3. Using this map, in (6.4.2), an agent particle takes likelihood value of the cell it is in.

Figure 6.4 shows result of the calibration for few of the sensitive parameters. As we can see, the distribution over parameter values have been refined. In Figure 6.4a, the uncertainty on

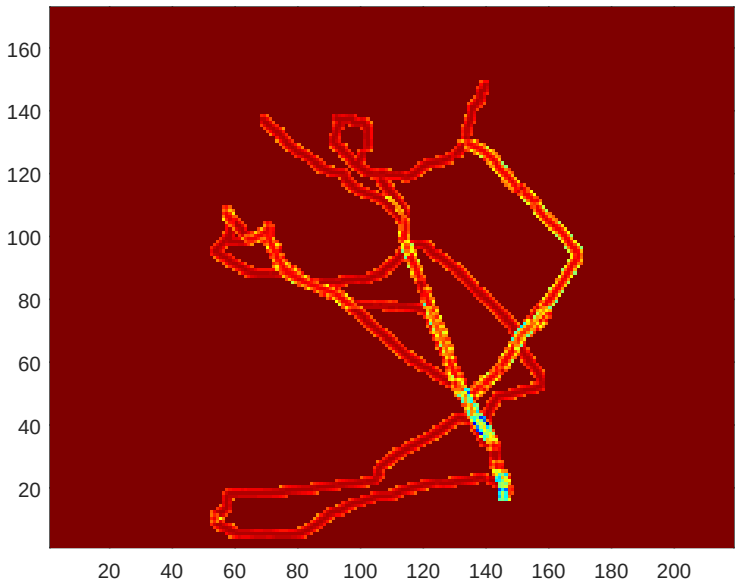
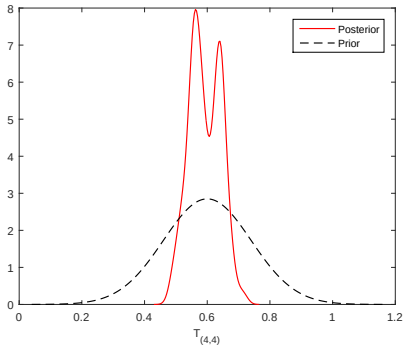
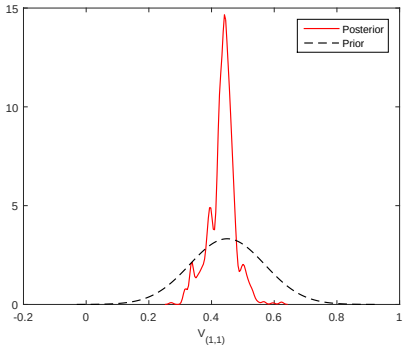


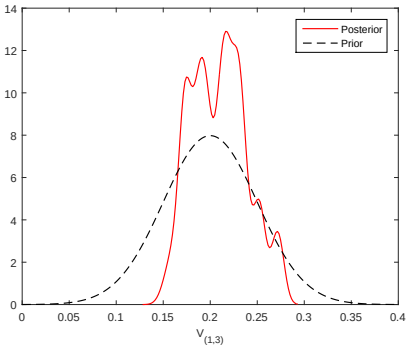
Figure 6.3: GPS log distribution in gridded environment.



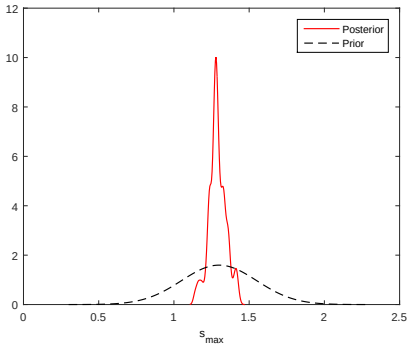
(a) $T_{(4,4)}$



(b) $V_{(1,1)}$



(c) $V_{(1,3)}$

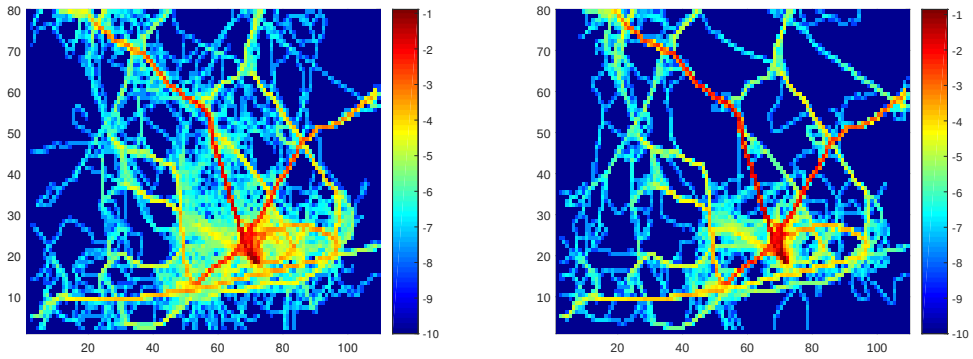


(d) s_{max}

Figure 6.4: Prior and posterior distributions over parameters. From the calibration results, mean and variance of distributions are used to acquire feature transition matrices.

Algorithm 9 GPS log distribution in gridded environment for computing model parameter likelihood.

$\mathbf{ModelParameterLikelihood}()$
h = Height of the gridded search area
w = Width of the gridded search area
m = Size of grid cell
$\mathbf{A} = 0 : |\mathbf{A}| = h \times w$
for $i = 1$: Number of GPS logs **do**
  $\mathbf{B} = 0 : |\mathbf{B}| = h \times w$
  for $j = 1$: Number of points on i^{th} GPS log **do**
    $[x \;\; y] = \mathbf{z}^{(i)}_{log,j}$
    Set a_c to be the cell at coordinate x, y
    $lh = \int_{y-m}^{y+m} \int_{x-m}^{x+m} p\left(\mathbf{z}^{(i)}_{log,j} | x, y\right) p\left(x, y | a_c\right) d_x d_y$
    if $\mathbf{B}_{(x,y)} < lh$ **then**
      $\mathbf{B}_{(x,y)} = lh$
    end if
    for $n = 1 : 8$ **do**
      Set a_n to be the n^{th} neighbouring cell if a_c
      Set $[x_n \;\; y_n]$ to be the coordinate of a_n
      $lh = \int_{y_n-m}^{y_n+m} \int_{x_n-m}^{x_n+m} p\left(\mathbf{z}^{(i)}_{log,j} | x_n, y_n\right) p\left(x_n, y_n | a_l\right) d_{x_n} d_{y_n}$
      if $\mathbf{B}_{(x_n,y_n)} < lh$ **then**
        $\mathbf{B}_{(x_n,y_n)} = lh$
      end if
    end for
  end for
  $\mathbf{A} = \mathbf{A} + log(\mathbf{B})$
end for



(a) Initial distribution with non-calibrated agent

(b) Initial distribution with calibrated agent

Figure 6.5: These two plots show the log of initial distribution over theLP location using both non-calibrated and calibrated agents respectively.

$T_{(4,4)}$ is reduced. This transition denotes the probability that the LP, if on a path, will stay on that path. The optimised result agrees with anecdotal observations in the search literature that most LP's prefer paths and that they tend to follow linear features. It is also reported in search

Table 6.5: Calibrated values for transition matrix parameters.

Parameter	Value
$\mathbf{M}_{\mu}^{T}$	$\begin{bmatrix} 0.1065 & 0.1049 & 0.3328 & 0.4559 \\ 0.1038 & 0.0956 & 0.3551 & 0.4455 \\ 0.0506 & 0.0503 & 0.3425 & 0.5566 \\ 0.0502 & 0.0508 & 0.3064 & 0.5927 \end{bmatrix}$
$\mathbf{M}_{\sigma}^{T}$	$\begin{bmatrix} 0.0216 & 0.0208 & 0.0583 & 0.0601 \\ 0.0264 & 0.0192 & 0.0506 & 0.0550 \\ 0.0053 & 0.0062 & 0.0499 & 0.0485 \\ 0.0061 & 0.0059 & 0.0484 & 0.0493 \end{bmatrix}$
$\mathbf{M}_{\mu}^{V}$	$\begin{bmatrix} 0.4339 & 0.3663 & 0.1998 \\ 0.4549 & 0.3432 & 0.2019 \\ 0.4916 & 0.4084 & 0.1000 \end{bmatrix}$
$\mathbf{M}_{\sigma}^{V}$	$\begin{bmatrix} 0.0448 & 0.0492 & 0.0319 \\ 0.0588 & 0.0549 & 0.0289 \\ 0.0562 & 0.0581 & 0.0188 \end{bmatrix}$
$\mathbf{M}_{\mu}^{E}$	$\begin{bmatrix} 0.3505 & 0.2918 & 0.1559 & 0.2007 & 0.0010 \end{bmatrix}$
$\mathbf{M}_{\sigma}^{E}$	$\begin{bmatrix} 0.0515 & 0.0481 & 0.0284 & 0.03841 & 0.0001 \end{bmatrix}$

literature that lost people traverse the environment along routes that offer least resistance [3]. Therefore, they tend to stay in areas with the least amount of vegetation. This is in agreement with our calibrated result of $V_{(1,1)}$ in Figure 6.4b. However at times in reality the LP may want to transition from sparse to dense vegetation $V_{(1,3)}$ especially when trying to get to an intended target location or trying to change paths, that can result in going over dense vegetation, which has been the case in our data collection experiment. This is once again supported by our calibrated parameter shown in Figure 6.4. The calibrated distribution over s_{max} is also in agreement with the normal human walking speed. Although the prior over s_{max} had large variance, the calibrated distribution reduces the uncertainty to ranges of walking speed normally exhibited by young aged people which is between 1.2 to 1.5 meters per second. Reflecting the walking speed of data collection participants.

The mean and standard deviation of the Gaussian approximation to the calibrated parameters are given in Table 6.5.

6.5 Agent Model Verification and Validation

Verification and validation are essential to agent model development. The former, refers to the correctness of the internal structure and workings of a model [143]. The latter concerns how well model outcomes represent real world behaviour [171] i.e. in our case, how agent generated trails represent lost people movements in wilderness.

While importance of verification and validation have been acknowledged in literature [171, 172], due to the complexity of the two processes, they have largely been ignored or minimally

discussed in literature [99, 111]

Although verification and validation are usually considered independent tasks in model development, in reality, specifically with agent model development, they can be coupled. Robert et al. in [173] state that model verification and validation is a process and is part of the total model development process. Since determining the absolute validity of a model over its intended domain is often too costly and time consuming, it suggests conducting tests and evaluations until sufficient confidence is obtained and model can be considered valid for its intended application.

To achieve this, [173, 174] suggests three basic approaches to decide if a model is valid. Each requiring the verification and validation as part of the model development. These are:

- *Conceptual validity:* This relates to determining that the theories and assumptions underlying the conceptual model are correct and that the model representation of the target is reasonable for intended purpose of the model. It is achieved doing things like:

  - If a Markov chain is used, ensuring the system have the Markov property, and that the states and transitions probabilities are correct.
  - Evaluation of each sub-model and overall model to determine if they re reasonable and correct for the intended purpose of the model.
  - Determining parameter values from data or calibrating model parameters.

  The validation techniques mainly used for this are:

  - *Face validation*: Asking people knowledgeable about the conceptual model to determine if it is correct and reasonable for its purpose, which requires examining the flowchart or graphical models of the system.
  - *Traces*: To ensure the logic and rules used in the model are accurate and the agent behave as expected, the movement of the agent is followed during simulation to determine logic and rules are fired as expected..

  ,

- *Computerised model verification:* This relates to ensuring computer programming and implementation of the conceptual model is correct. This is done by checking that the development model is bug free and that the logic has been coded correctly. The main techniques used to achieve this are structured walk-through and tracing.

- *Operational validity:* This determines if a model's output behaviour has sufficient accuracy for the model's intended purpose over domain of the model's intended applicability. There are quite few techniques listed in [173] to test for this. The mains ones are parameter *sensitivity analysis* and *historic data validation*, the later of which means using part of real-word data to build the model and the remaining part to test it.

Another similar approach is proposed in [144]. Here too, considering the non-linearity of the agent model, and availability of real world data, a five stage validation process is proposed, which are: *Conceptual validity*, *face validation*, *sensitivity analysis*, *calibration* and *statistical validation*. This basically covers what was listed before, but in different order.

In this research, following these validation frameworks, while we have performed rigorous Conceptual validity and computerised model verification by:

- Reviewing the mathematical model developed ensuring they are sound.
- Evaluating agent model behaviour both by part (Section 4.5 and Section 5.3) and as a whole (for local agent model in Section 4.7 and global agent model in Section 5.5).
- Performing sensitivity analysis and calibrating the agent model parameter values.
- Testing the code both at unit level and at whole model level ensuring the code is bug free and covers all the logic and rules to a satisfactory level.

we have only been able to perform limited operational validity of our proposed agent model. We have not been able to perform historic data validation or statistical validation.

Ideally, we should have access to large set of logs of lost people movements in wilderness and it should be illustrated that the agent model is able to generate trails that faithfully represent lost peoples movements. However, in our case, it is very difficult to capture real data of lost people movement. Even, if data is captured, there are caveats in validating and analysing output of a model like ours using real world observations [82]. For example:

- Both the agent model and the target movements are stochastic (that is, based partly on random factors). Thus comparison between the model output and real world data are unlikely to correspond on every occasion.
- The outcomes depend on the precise initial conditions chosen because these affect the history of the simulation. In other words, the outcomes may be very sensitive to the precise values of some of the assumptions in the model i.e. initial conditions, stochastic behaviour/ interaction with environment.

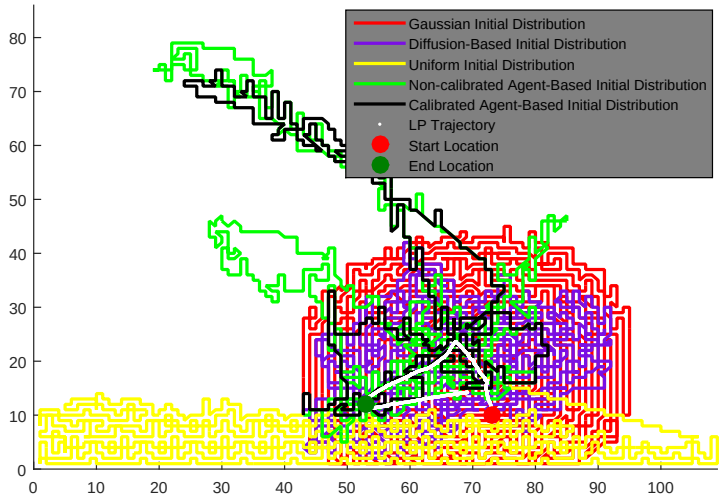


Figure 6.6: The UAV search path using the different initial distribution types.

- Even if the results obtained from the simulation match those from the target, there may be some aspects of the target that the model cannot reproduce.

For such complex models, the questions of validity and of verification are hard to distinguish [82], which reflects what we said at the start of this section.

Having said this, we can indirectly validate the agent model and show that since it is calibrated using real data, it is a faithful representation of the LP movement in wilderness. We do this by comparing the performance UAV assisted search using non-calibrated and calibrated agent generated initial distributions. Figure 6.5 shows the difference in the generated initial distribution using non-calibrated and calibrated agent models. As can be seen, the distribution generated using the calibrated agent model is more refined than the one generated using the non-calibrated agent model. We believe the former will allow for better hypotheses of LP whereabouts and efficient search resource allocation.

To test this hypothesis, we repeated the second experiment in Section 5.5 and performed the search using an initial distribution generated using the calibrated agent model.

To illustrate the effect calibrated agent generated trails has on search, we have overlaid the UAV search path on search paths of experiment in Section 5.5 illustrated in Figure 6.6. Results of search for four target end locations randomly selected along the four GPS logs is given in Table 6.6. Comparing the values in table to values in Table 5.5, we can see that, using the calibrated agent model, search times were significantly improved halving the times taken by non-calibrated agent model.

While this does not represent statistical validation of the agent model, it does improve the

Table 6.6: The time-to-locate factor using calibrated agent model for the four different target end locations.

Trial	1	2	3	4
Calibrated agent mode	648	1154	102	315

plausibility of the agent model developed to represent movement of lost people in wilderness on the condition that it is well calibrated.

6.6 Summary

In this chapter we investigated parameter tuning. We argued that parameter calibration can be very difficult when dealing with models that consist of many parameters such as ours.

To address this issue, we decomposed the calibration challenge into two steps: pre-calibration step of parameter screening and the calibration step. In the parameter screening step, we used a fractional factorial Design Of Experiment (DOE) to identify sensitive (or important parameters). Then in the parameter calibration step, we calibrated the identified sensitive parameters using GPS logs of peoples movement in a wilderness like area. The calibration was done using SMC^2. Of course, the quality of the calibration depends on the underlying observed data. In our case, we used 8 GPS logs of peoples' movement in a wilderness like area. We understand that the logs may not be accurate representation of lost people movement in wilderness. But, we have shown that the agent model parameters can be calibrated, even if they are many. We showed that by performing sensitivity analysis, the model parameters that are very sensitive to small perturbations in their value can be identified, which can then be used in the calibration process reducing the computational complexity of calibrating the complete agent model.

Having calibrated the agent model parameters, we needed to show that the agent model developed was able to represent actual lost peoples' movements and interactions with environment. To do this, we detailed a couple of validation frameworks. Each made up of a number of phases. We argued that, while we had performed most of the steps in the validation phases in each of the framework during the development of the agent model, we could only do indirect validation against observed data due to difficulty of getting observed data.

We illustrate that in comparison to using an initial distribution generated using the non-calibrated agent model, search time can be reduced by using an initial distribution generated using the calibrated agent model. To do this we repeated the second experiment in previous chapter . We interpret this to mean that the calibrated agent model was able to better represent movement of data collection experiment participants compared to non-calibrated agent model.

As a result, the UAV was able to identify the path taken by the target much quicker.

This far we have only considered a simplistic update model- only considering the LP observation. To make better predictions of LP whereabouts using our proposed search framework, we need to consider other observation types such new observations of the land cover, observations of the evidence deposited by lost people in the update phase. To do this, we need an improved update model.

Chapter 7

Lost Person Trail Inference Using Evidence and Land Cover Classification

7.1 Introduction

This chapter addresses the second issue highlighted in Chapter 3: how to exploit evidence and dynamic updates of land cover classification information. No previous search algorithm has attempted to use these types of indirect information before.

We start by presenting this update model in Section 7.2. The implementation details of the update model is decomposed into two parts. The first part, in Section 7.3, derives the equations to exploit evidence deposited by the LP. Section 7.4 illustrates the use of this information in a search scenario, and shows that search times can be reduced by a factor of up to 25. Section 7.5 considers the second part in which we incorporate land cover classification information. This model is evaluated in Section 7.6. We show that in the absence of evidence observation, just considering land cover changes in the update phase can help reduce search times by a factor of 3. Finally, we present the summary of the chapter in Section 7.7.

Key terms used in this chapter

- **Initial distribution:** This refers to the initial distribution over LP trail generated before search operation is initiated.
- **Prior distribution:** This refers to the distribution over LP trail at time-step $k-1$.
- **Posterior distribution:** This refers to the updated distribution over LP trail at time-step k.

7.2 Probabilistic Model of Search

As mentioned in Section 3.4.3, search is performed using recursive predict and update steps. We will cover each step separately but in reverse order for coherency. First, we will present

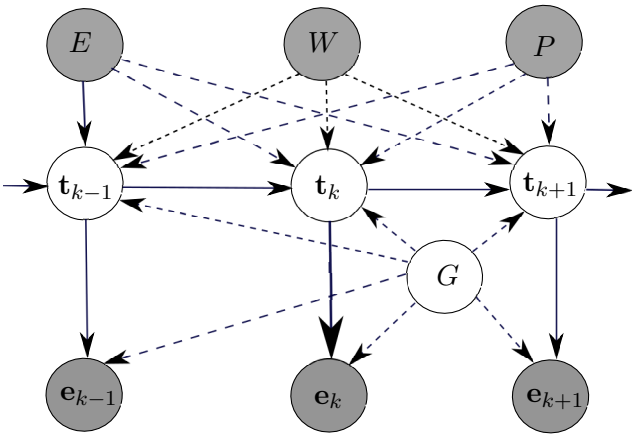


Figure 7.1: The graphical model showing the evidence deposition as a function of the LP's movement. The LP's trail state at time k, $\mathbf{t}_k$, is determined by the weather W, the land cover (E and G) and the profile P. Filled and clear circles represent observed and latent variables.

the update step considering observations of the LP and evidence and assume that the land cover classification used for initial distribution is current. Then will well present the predict step, where we will model affects of new land cover classification on the agent trails.

Throughout we assume that:

- The search platform flies at a constant altitude, with a fixed camera frame and dynamics constraint to a single cell movement in a time-step directed by the term $p\left(\mathbf{u}_k|\mathbf{T}, \mathbf{x}_{k-1}\right)$ in (3.16).

- Environment observations (detected evidence features or the LP) are well localised i.e. we know the exact coordinates.

However, before going to the details of the update models, we decompose the graphical model in Figure 3.4 and present two sub models showing time dependency of system variables, describing the deposition of evidence and UAV observations.

7.2.1 Graphical Models of Evidence Deposition and UAV Observation

7.2.1.1 Evidence Deposition Graphical Model

The graphical model of LP movement and the deposition of evidence is shown in Figure 7.1. This model captures the fact that at each step, there is a probability that the LP can drop something or cause a change such as footprint. Evidence is deposited along the trail, which is strongly controlled by the LP's interaction with the terrain, the weather W and the profile P.

7.2.1.2 UAV Movement and UAV Observations Graphical Model

The graphical model of UAV's observation is shown in Figure 7.2. The UAV is driven by a sequence of control inputs. At each time-step, the UAV receives a set of observations which are

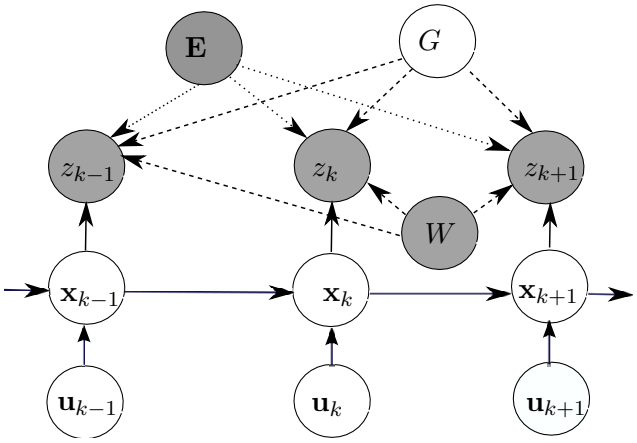


Figure 7.2: The graphical model for the Unmanned Aerial Vehicle movement and observation. The observation z_k at time k is a function of the UAV state $\mathbf{x}_k$, the set of evidence features $\mathbf{E}$, the land cover G (which affects the visibility of the evidence, and introduces non-evidence features $\mathbf{N}$) and the weather W (which affects visibility). The movement of the UAV at time-step k is directed by control inputs $\mathbf{u}_k$. The filled and clear circles represent observed and latent variables respectively.

a function of evidence deposited, land cover type G, and the weather conditions W. For example, footprints cannot be readily spotted using a camera-equipped UAV from the air. Weather can affects target visibility by, for example, reducing visibility. Conversely, lighting in the early morning can enhance target visibility [175].

7.3 Update Using Lost Person and Evidence Observations

Recall that $p(\mathbf{T}_B|\Lambda) = \{\mathbf{T}_B^{(i)}, w_B^{(t,i)}\}_{i=1}^N$, where $\mathbf{T}_B^{(i)}$ is the i^{th} agent trail particle with associated weight $w_k^{(t,i)}$. The update is performed by revising this weight.

We consider two types of information. Defining current cell being observed to be a_c, the first type, which has been widely used, is to use direct observations of the LP, which provides information about the LP's location. Using this information, we will update agent trails that does — or does not — end in a_c. The second type, which is novel in this research, is to explore the evidence related information, which tells us if the LP has passed through the cell a_c. Using this information, we will updated agent trails that does — or does not — pass through a_c.

7.3.1 Lost Person Detection Model

To incorporate observations of the LP in to the update model, we use an approach similar to the grid-based update in Section 2.3: we define a person detection model that considers both *false alarms* and *missed detections*. Defining $z_k^p \in \{0, 1\}$ to be the binary random variable which shows if a person is present (1) or not (0), and $\mathbf{P} = \{G, W, \mathbf{x}_k\}$, the person detection model is

$$p\left(z_k^p|\mathbf{t}_S,\mathbf{P}\right)=\begin{cases}p\left(z_k^p=0|\mathbf{t}_S\in a_c,\mathbf{P}\right) & =\beta_p^{(a_c)}\\ p\left(z_k^p=1|\mathbf{t}_S\in a_c,\mathbf{P}\right) & =1-\beta_p^{(a_c)}\\ p\left(z_k^p=0|\mathbf{t}_S\notin a_c,\mathbf{P}\right) & =1-\alpha_p^{(a_c)}\\ p\left(z_k^p=1|\mathbf{t}_S\notin a_c,\mathbf{P}\right) & =\alpha_p^{(a_c)}\end{cases}, \tag{7.1}$$

where $p\left(z_k^p=1|\mathbf{t}_S\notin a_c,\mathbf{P}\right)$ models false alarm and $p\left(z_k^p=0|\mathbf{t}_S\in a_c,\mathbf{P}\right)$ models missed detection with detection probabilities $\alpha_p^{(a_c)}$ and $\beta_p^{(a_c)}$ respectively. These error rates quantify the noise characteristics of a sensor. For a given sensor both $\alpha_p^{(a_c)}$ and $\beta_p^{(a_c)}$ can be determined experimentally or by sensor specifications [17, 18].

7.3.2 Evidence Detection Model

Similar to the person detection model, we need to define an evidence detection model to incorporate observations of the evidence. A crucial difference is that evidence detection is composed of two events: The event that evidence is detected and the event that the evidence was actually deposited by the LP. Therefore, defining $\mathbf{S}=\{G,\mathbf{E},W,P,\mathbf{x}_k\}$, e_k to be the evidence deposition event at time-step k, and $z_k^e\in\{0,1\}$ to the binary random variable which shows if an evidence feature is present (1) or not (0),

$$p\left(z_k^e=1|\mathbf{T},\mathbf{S}\right)=\sum_{e_k}p\left(z_k^e=1|e_k\in a_c,\mathbf{S}\right)p\left(e_k\in a_c|\mathbf{T},\mathbf{S}\right), \tag{7.2}$$

where the first term on the right models the likelihood of detecting the evidence feature given it is deposited and the second term models probability of it actually being deposited by the LP.

Since an LP can only deposit or leave evidence in a location if they have traversed it, this event has non-zero probability if the LP passes through a_c. Defining $\beta_e^{(a_c)}=1-p\left(z_k^e=1|a_c\in\mathbf{T},\mathbf{S}\right)$, the evidence detection model is given by

$$p\left(z_k^e|\mathbf{T},\mathbf{S}\right)=\begin{cases}p\left(z_k^e=0|a_c\in\mathbf{T},\mathbf{S}\right) & =\beta_e^{(a_c)}\\ p\left(z_k^e=1|a_c\in\mathbf{T},\mathbf{S}\right) & =1-\beta_e^{(a_c)}\\ p\left(z_k^e=0|a_c\notin\mathbf{T},\mathbf{S}\right) & =1-\alpha_e^{(a_c)}\\ p\left(z_k^e=1|a_c\notin\mathbf{T},\mathbf{S}\right) & =\alpha_e^{(a_c)}\end{cases}, \tag{7.3}$$

where $p\left(z_k^e=0|a_c\in\mathbf{T},\mathbf{S}\right)$ and $p\left(z_k^e=1|a_c\notin\mathbf{T},\mathbf{S}\right)$ model missed detection and false detection respectively.

7.3.3 Update Model Using Lost Person and Evidence Observations

With the LP and evidence detection models defined, the weight on each agent trail particle is updated using Bayes' rule (3.17). Assuming known platform location and current land cover classification, update can take one of two forms: if LP is detected in a cell

$$w_k^{(t,i)} = \eta \cdot p\left(z_k^p = 1|\mathbf{t}_S^{(i)}, \mathbf{P}\right) w_{k-1}^{(t,i)}, \tag{7.4}$$

where $w_{k-1}^{(t,i)}$ is the weight of the trail particle given observations up until time-step $k-1$, and η is the constant normalising term. The value of $p\left(z_k^p = 1|\mathbf{t}_S^{(i)}, \mathbf{P}\right)$ depends on whether $\mathbf{t}_S^{(i)} \in a_c$ or $\mathbf{t}_S^{(i)} \notin a_c$.

However, if LP is not detected, the update is done by

$$w_k^{(t,i)} = \eta \cdot p\left(z_k^p = 0|\mathbf{t}_S^{(i)}, \mathbf{P}\right) p\left(z_k^e|\mathbf{T}^{(i)}, \mathbf{S}\right) w_{k-1}^{(t,i)}. \tag{7.5}$$

The value of $p\left(z_k^p = 0|\mathbf{t}_S^{(i)}, \mathbf{P}\right)$ depends on whether $\mathbf{t}_S^{(i)} \in a_c$ or $\mathbf{t}_S^{(i)} \notin a_c$. Similarly, the value of $p\left(z_k^e|\mathbf{T}^{(i)}, \mathbf{S}\right)$ depends on whether $z_k^e = 1$ or $z_k^e = 0$ and $a_c \in \mathbf{T}^{(i)}$ or $a_c \notin \mathbf{T}^{(i)}$.

A known issue with particles as mentioned in Section 3.3.1 is particle degeneracy. To counter this, we need to define a particle re-sampling strategy, which takes into account the cause of degeneracy.

7.3.4 Agent Particle Re-sampling Strategy

To avoid particle degeneracy, after every update step, it is determined if re-sampling is required. This is done by computing N_{eff} using (3.7) and checking if $N_{eff} < \alpha N$, where $\alpha = 0.6$.

In the re-sampling step, the agent trail indices are sampled from the set $\left\{\mathbf{T}^{(i)}, w_k^{(t,i)}\right\}_{i=1}^N$ using

$$\left[\{j\}_{j=1}^N, i^j\right] \sim RESAMPLE\left[\left\{\mathbf{T}^{(i)}, w_k^{(t,i)}\right\}_{i=1}^N\right], \tag{7.6}$$

where i is the index of the re-sampled particle, which is a copy of the j^{th} particle in the original distribution. From here on, we will call the samples selected in the re-sampling step as *parent particles* and their clones as *child particles*. The re-sampling in our case is done using stratified re-sampling method [176] produces lower variance than simple random re-sampling After re-sampling the agent particles, two processes take place.

First, all agent particles weights are normalised,

$$w_k^{(t,i)} = \frac{w_k^{(t,i)}}{\sum_{i=1}^N w_k^{(t,i)}}, \tag{7.7}$$

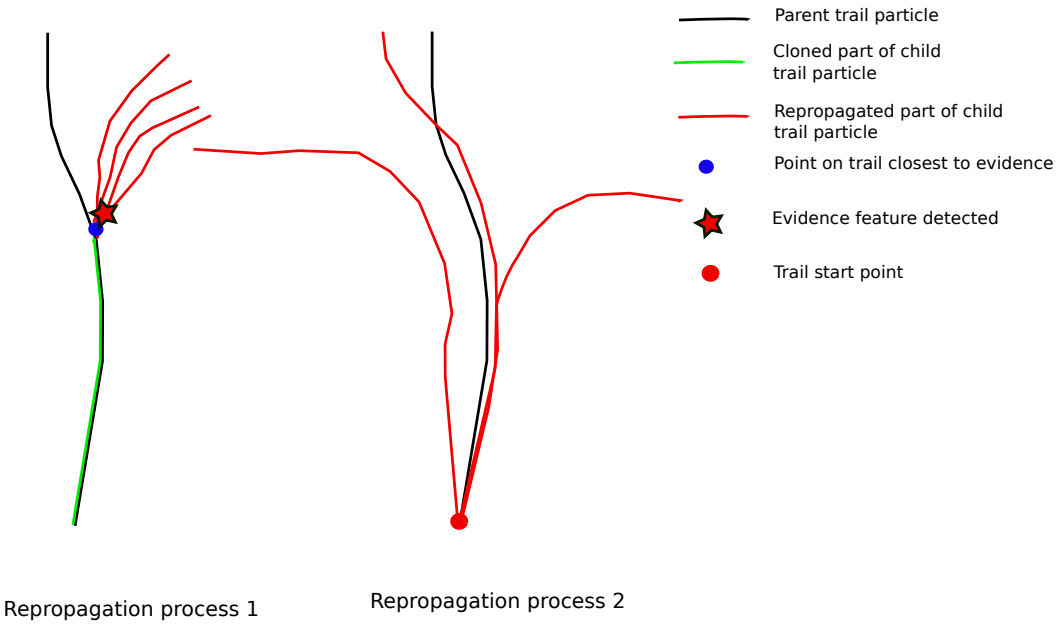


Figure 7.3: Trail re-propagation processes. The blue and red circles represent $\mathbf{t}_\tau$ and $\mathbf{t}_L$ respectively.

ensuring $\sum_{j=1}^{N} w_k^{(t,i)} = 1$.

Second, the child particles are propagated. Depending on the cause for the re-sampling, propagation is done by using on of the following two processes illustrated in Figure 7.3:

- *Propagation Process 1:* If the re-sampling is triggered by positive evidence observation, to use the detected evidence as a secondary PLS, all child particles are propagated forward in time from the point closest to the detected evidence.

  Let $\mathbf{t}_\tau^{(j)}$ to be the point on $\mathbf{t}^{(j)}$ closest to the detected evidence determined by using Euclidean distance. First, each child particle is cloned (including its memory until time-step τ) by a clone operator

$$\mathbf{t}_{L:\tau}^{(i)} = \left\{ \mathbf{t}_L^{(j)}, \mathbf{t}_{\tau=1}^{(j)}, \cdots, \mathbf{t}_\tau^{(j)} \right\}. \tag{7.8}$$

  Then, the cloned child particle is propagated forward in time to time-step S using a variant of (3.3) given as

$$p(\mathbf{T}^{(i)}|\mathbf{t}_{L:\tau}^{(i)}, \Lambda) = p\left(\mathbf{t}_{L:\tau}^{(i)}|\Lambda\right) \prod_{k=\tau+1}^{S} p\left(\mathbf{t}_k^{(i)}|\mathbf{t}_{k-1}^{(i)}, \Lambda\right), \tag{7.9}$$

  where the first term on the right hand side is the part of agent trail cloned by (7.8), and the second term models the re-propagation of the agent sample through the environment from time-step τ to time-step S.

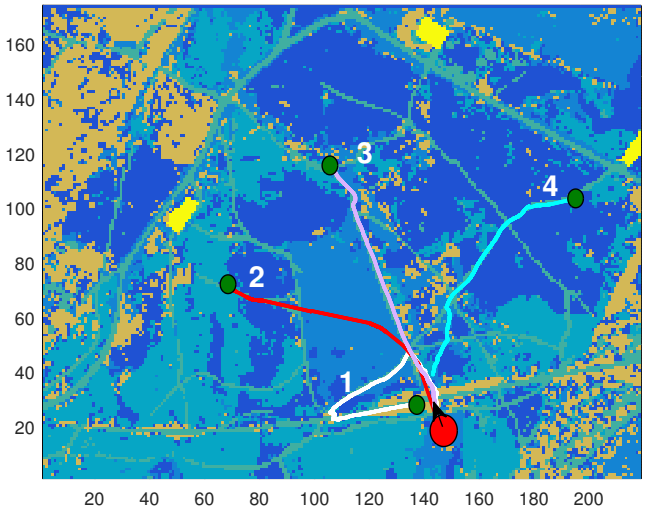


Figure 7.4: Movement logs of four people, with red circle marking PLS. Green circles represent the actual end location.

- *Propagation Process 2:* If the re-sampling is not triggered by evidence observation, then child particles are initialised by

$$\mathbf{t}_L^{(i)} = \mathbf{t}_L^{(j)}, \tag{7.10}$$

and then propagated by (3.3).

7.4 Evaluation of Lost Person and Evidence Update Models

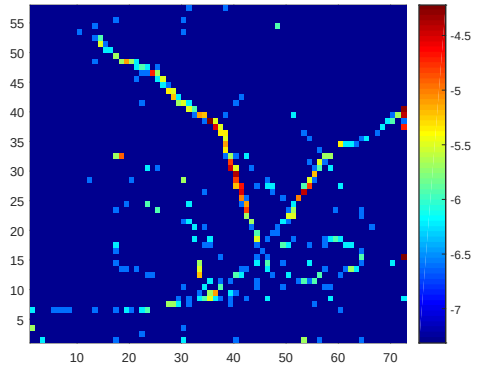
The aim of this experiment is to investigate the effect of using evidence information. We do this by comparing search results of our search model with the grid-based search model described in Chapter 2.

Consider the scenario in Figure 7.4 with movement logs of four different people reported lost. In each case, the PLS corresponds the area marked by red circle. Each person was last seen walking in the direction of the arrow shown in the figure.

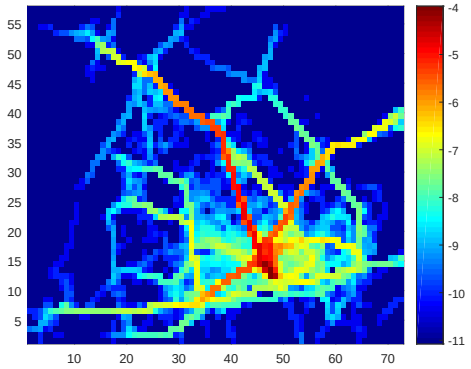
We start by generating two types of initial distribution: using the agent model developed in Chapters 4 and 5; and the diffusion model detailed in Chapter 2. The resulting distributions are shown in Figure 7.5. Since the LP cases are in the same area, we use the same initial distribution in all search cases.

7.4.1 Environment Setup

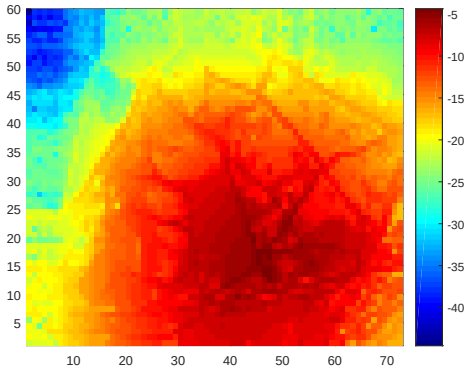
This experiment is performed using the New Forest data sets shown in Figure 4.8. The search area size is 800m $\times$ 1100m, discretised into equal cells of 5m on a side. The UAV starts its search from the PLS.



(a) Agent-based initial distribution on LP location



(b) Agent-based initial distribution on LP trail



(c) Diffusion-based initial distribution

Figure 7.5: The two figures in the top shows the generated initial distribution using the agent model. The figure in the bottom row show the initial distribution generated using the diffusion model. Both the agent model and the diffusion model use calibrated parameter values and are given in log. The axes on the right is map of colour intensity to log value

7.4.2 Lost Person Movement Model

For both agent model and diffusion model, we use the setup in Section 5.5.3 with the calibrated parameters in Table 6.5.

7.4.3 Unmanned Aerial Vehicle Path Planning

We use the look-ahead path planning method detailed in Section 2.3.6 with $w = 5$. The sensor detection model parameter values are given in Table 7.1. The high false evidence detection demonstrates that we are uncertain about it. For simplicity, we assume a constant value for $p\left(e_k \in a_c | \mathbf{T}, \mathbf{S}\right)$. In reality however, the value for this term would change depending on the environment state at area being observed. It is assumed that the throughout the search, the UAV flies at a constant altitude with a fixed size camera / sensor footprint, which is $(15m)^2$. This is reflected in the grid size of distributions in Figure 7.5.

Table 7.1: Detection model parameter values used for the update model.

Detection Parameters	Terms	value
Probability of evidence detection	$p\left(z_k^e = 1 \middle\| e_k \in a_c, \mathbf{S}\right)$	0.6
Probability of false evidence detection	$\alpha_e^{(a_c)}$	0.3
Probability of depositing evidence	$p\left(e_k \in a_c \middle\| \mathbf{T}, \mathbf{S}\right)$	0.6
Probability of LP missed detection	$\beta_p^{(a_c)}$	0.3
Probability of LP false detection	$\alpha_p^{(a_c)}$	0.2

7.4.4 Evaluation Metrics

To compare the performance of the two methods, we use two measure of performance: *time-to-locate* and *entropy* of the distribution over LP end location. We use the latter to evaluate the uncertainty in the LP's end location and how it evolves over time. High entropy represents, high uncertainty and reduced predictability of where they can be found, In contrast, low entropy represents low uncertainty and increased predictability. It is given by

$$H_k(\mathbf{t}_S) = -\sum_{i=1}^{M} p\left(\mathbf{t}_S \in a_i | Z_k, \Lambda\right) \log p\left(\mathbf{t}_S \in a_i | Z_k, \Lambda\right).$$

where $p\left(\mathbf{t}_S \in a_i | Z_k, \Lambda\right)$ is computed using (3.11).

7.4.5 Evidence and Lost Person Update Results

We compare the results of search using both agent and diffusion generated initial distributions using four different configurations:

- *Configuration 1 - Using the trail-based search model with LP but no evidence information:* This is performed only once for each scenario. In this configurations, the UAV path is computed using the evolving distribution over trail of the LP.

- *Configuration 2 - Using the proposed search model with LP and evidence information:* This is repeated 5 times for each scenario, each time with a random configuration of a maximum of 3 evidence deposited. In this configurations, the UAV path is computed using the evolving distribution over trail of the LP. Using this setting, we investigate the effects of different evidence configurations on search time.

- *Configuration 3 - Using grid-based search with initial distribution over LP end location generated using the agent model:* This is performed only once for each scenario. In this configurations the UAV path is computed using the evolving distribution over the end location of the LP generated using the agent model.

- *Configuration 4 - Using grid-based search initial distribution over LP end location generated using the diffusion model:* This is performed only once for each scenario. In this configurations the UAV path is computed using the evolving distribution over the end location of the LP generated using the diffusion model.

Figure 7.6 illustrates the evolving distribution over the LP trail (trail 2) using configuration 2. As can be seen, using the initial distribution, the UAV has managed to search the environment, updating the initial distribution with observations of the LP and evidence and finally locate the LP and infer a trail likely to have been taken by the LP. Figure 7.7 illustrates the case during the search, when as a result of positive evidence observation, re-sampling is performed. As can be seen, using the first propagation process detailed in Section 7.3.4, the evidence location serves as a kind of secondary PLS from where majority of the new particles spawn out. An added advantage of using trail-based distribution to infer the LPs whereabouts is that more than one potential trail can be inferred to have been taken by the LP, as illustrated in Figure 7.8. This does two things: First, it helps identify the motivation of the LP for ending where found. Second, it provides the ground rescue team with alternatives routes of getting to the LP.

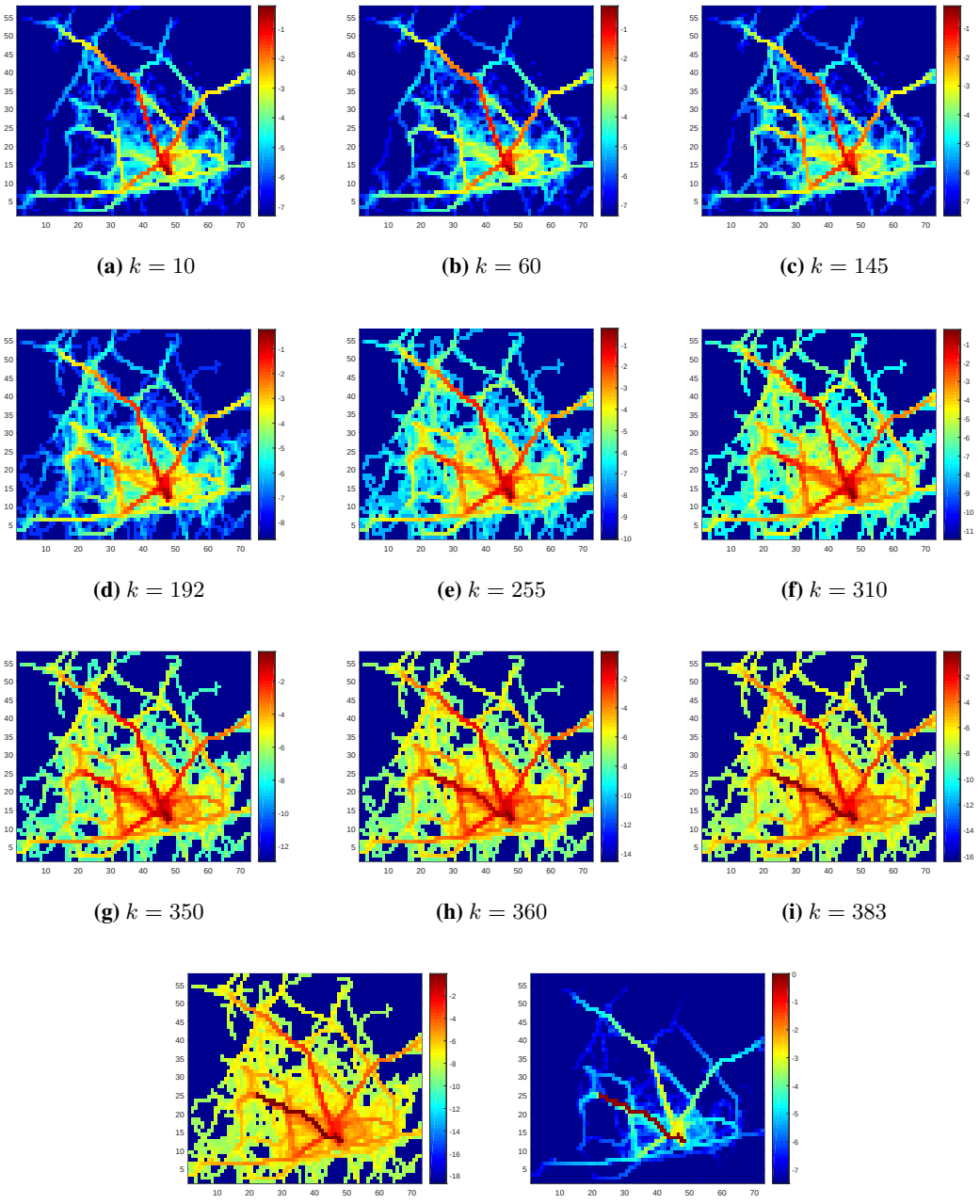
Figures 7.9 and 7.10 illustrate the entropy for the LP search cases using the above search configurations.

As can be seen, our search model has clearly out-performed the normal gird-based search. We will present the analysis of performance for each search configuration separately:

- *Configuration 1:* The UAV was able to out-perform configuration 4. We believe the reason for this is the continuous re-computation of the UAV search path using the evolving distribution over the trail of the LP i.e. taking advantage of the conditional dependency of the cells in the gridded representation of the search environment.

- *Configuration 2:* In most cases, the UAV was able to locate the LP much quicker compared to other search configurations, especially compared to configuration 4. In few cases however, the search times were very close or higher than configuration 1 and 3. Upon inspection, it was found that in these cases, the evidence observed and considered in the update were very close to PLS. This resulted in the update of agent trails that branched to different directions, which in return resulted in long UAV search paths. In cases where the evidence was detected away from the PLS, our search model was able to direct the UAV to the location of the LP much quicker.

- *Configuration 3:* This configuration too out-performs configuration 4. We believe the reason for this too is that the agent model considers LP trail in generating the distribution



(a) $k = 10$ **(b)** $k = 60$ **(c)** $k = 145$

(d) $k = 192$ **(e)** $k = 255$ **(f)** $k = 310$

(g) $k = 350$ **(h)** $k = 360$ **(i)** $k = 383$

(j) $k = 385$ **(k)** $k = 385$

Figure 7.6: Evolving distribution over the LP trail (trail 2) using the LP and evidence observations. The distributions are given in log where high intensity colours represent high probability and low intensity colours represent low probability. Although the colour scheme is same across all figures, the scales are different (the right axis) capturing decreasing log values of areas deemed not to have been traversed by the LP. To illustrate the difference, the distribution at $k = 385$ is given twice, once with axes same as distribution at $k = 1$.

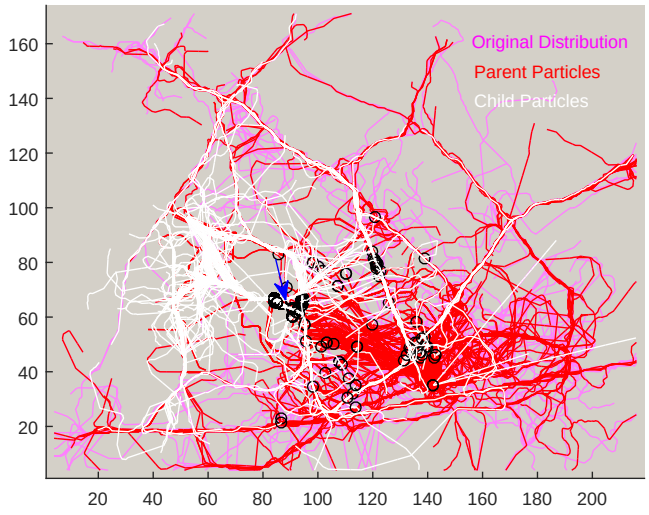


Figure 7.7: Re-sampled agent trails at $k = 192$. The black circles represents the point closest to the evidence detected along trail samples. The blue arrow indicates where the evidence is detected.

over the LP end location.

We can see in Figure 7.10 that with our search model, the distribution changes quite significantly whenever an observation of the evidence takes place (hence the sudden drops in entropy), which in case of grid-based search model only happens when the LP is observed.

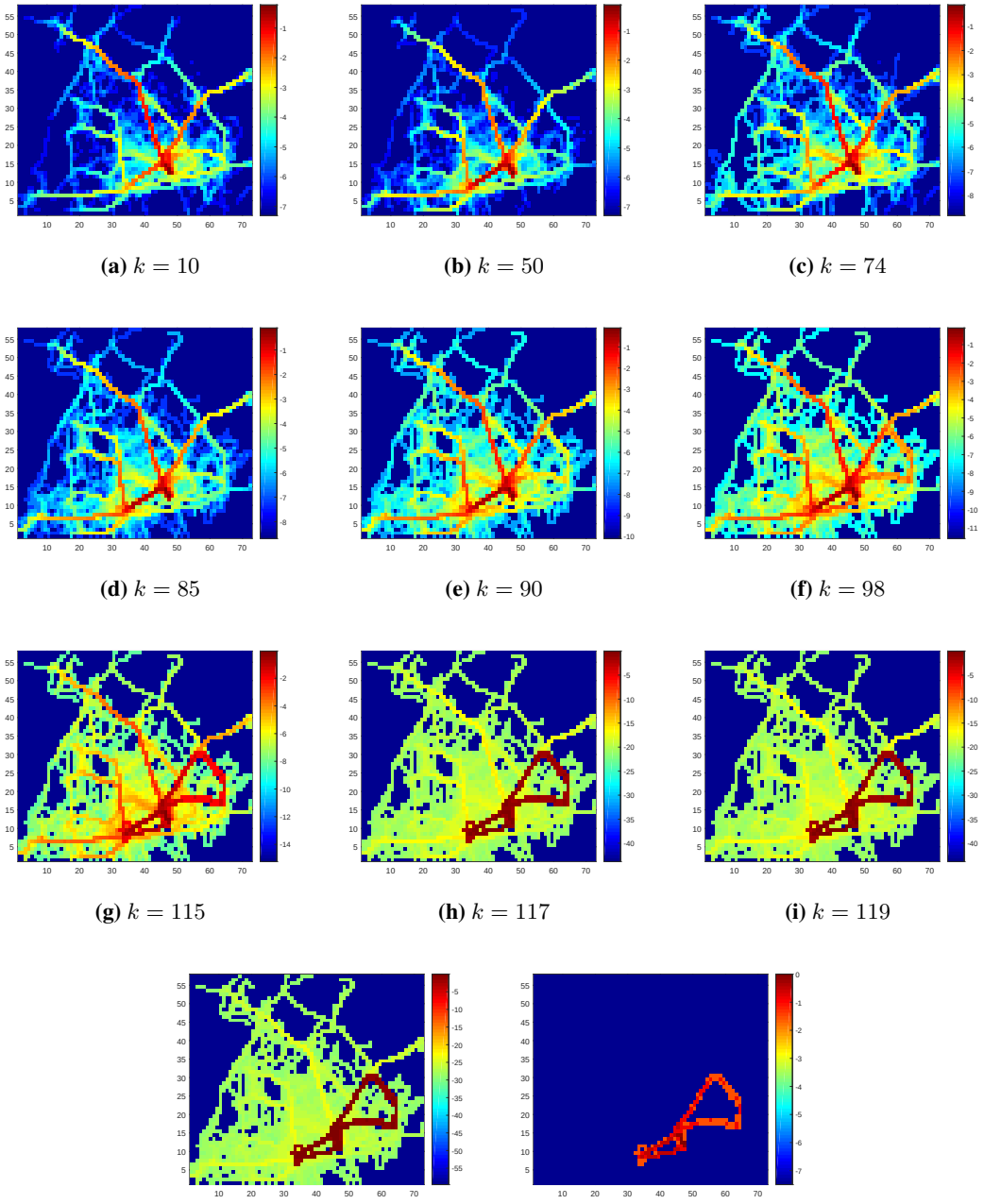
Overall, the superiority of our search model can be determined by the comparing the search times using the first three search configurations and configuration 4, which are:

- Configuration 1: Search time can improve by factor 3 to 5.
- Configuration 2: Search times can improve by factor of 2 to 25.
- Configuration 3: Search times can improve by factor of 0.25 to 5.

7.5 Update Model Using Land Cover Classification

The aim of the land cover classification update is to improve the land cover representation. As mentioned in Chapter 2, LPs traverse the environment by interacting with it and performing re-orientation depending on precept of it, which means their decision making is affected by the features they encounters.

Land cover classification can be performed by methods such as a maximum Likelihood Classifier (MLC) [25, 28], which can be used to estimate the probability of a pixel observed represented by a vector of spectral values belonging to feature class. Since land cover classification is covered by our affiliate Timothy Patterson, we will not cover it in this research.



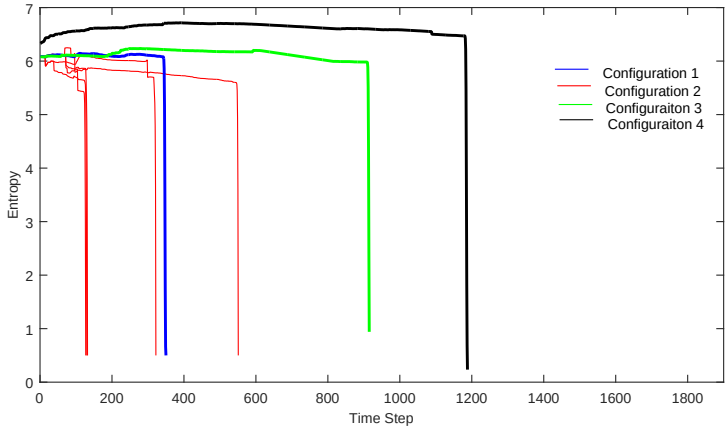
(a) $k = 10$ **(b)** $k = 50$ **(c)** $k = 74$

(d) $k = 85$ **(e)** $k = 90$ **(f)** $k = 98$

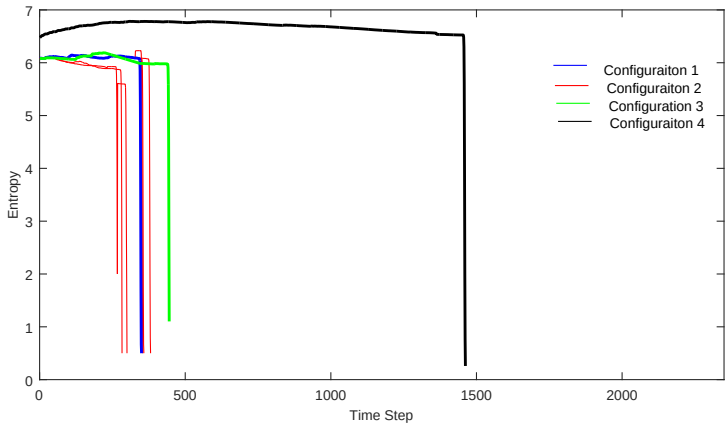
(g) $k = 115$ **(h)** $k = 117$ **(i)** $k = 119$

(j) $k = 121$ **(k)** $k = 121$

Figure 7.8: Evolving distribution over the LP trail (trail 1) using the LP and evidence observations. The distributions are given in log where high intensity colours represent high probability and low intensity colours represent low probability. Although the colour scheme is same across all figures, the scales are different (the right axis) capturing decreasing log values of areas deemed not to contain the LP or not to have been traversed by the LP. To illustrate the difference, the distribution at $k = 121$ is given twice, once with axes same as distribution at $k = 1$.



(a) Trail 1



(b) Trail 2

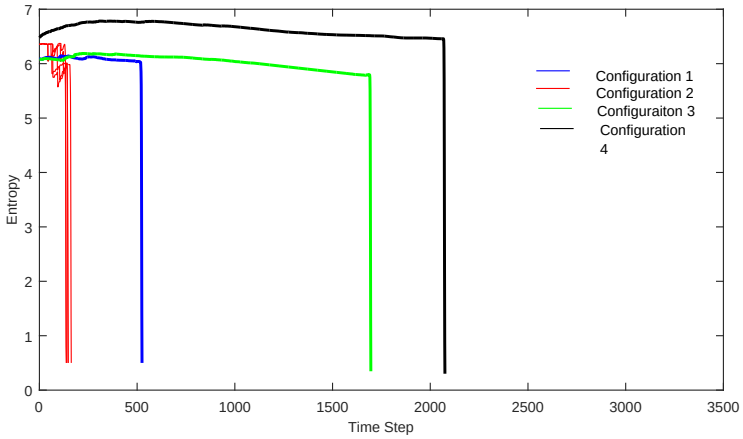
Figure 7.9: Comparison of the entropies computed during the life of search using the proposed search model and grid-based search models. Because agent trail-based search with evidence considered was repeated five times for each search case, each with a random configuration of evidence along the trail, there are five entropies in red.

Incorporating new classification of a cell in the search area requires re-propagating trajectories affected by it. For this we need a re-propagation strategy.

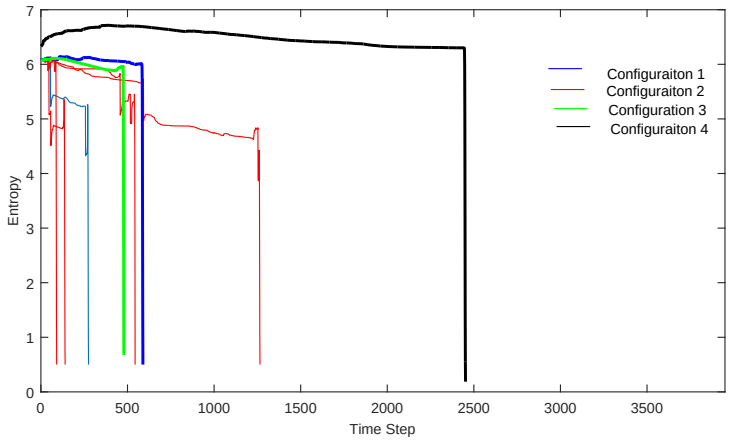
7.5.1 Land Cover Classification-Based Agent Trail Re-Propagation Strategy

To model affect of land cover re-classification on the agent trails, each cell in the discretised representation of the environment is treated as an agent itself with the sole purpose of remembering the agent trail IDs and state at the time of first interaction. Using this information, all agent trails affected by the re-classified cell are re-propagated from the point they first interacted with the cell re-classified.

Considering the cell being observed at time k to be a_c, the term $p\left(\mathbf{T}|G, \mathbf{x}_{k-1}, \Lambda\right)$ in (3.16) re-propagates all particles affected by a_c from the point of fist interaction. Assuming that the



(a) Trail 3



(b) Trail 4

Figure 7.10: Comparison of the entropies computed during the life of search for the LP using the proposed search model and grid-based search models.

i^{th} agent trail first interacted with a_c at time-step τ, agent particles propagation is given by

$$p(\mathbf{T}^{(i)}|\mathbf{t}^{(i)}_{L:\tau}, \Lambda) = p\left(\mathbf{t}^{(i)}_{L:\tau}|\Lambda\right) \prod_{k=\tau+1}^{S} p\left(\mathbf{t}^{(i)}_{k}|\mathbf{t}^{(i)}_{k-1}, \Lambda\right) \tag{7.11}$$

where $p\left(\mathbf{t}^{(i)}_{L:\tau}|\Lambda\right)$ is the history of the trail up until time-step τ and the second term propagates the agent trail further from time-step τ to S using the latest land cover representation.

This process is illustrated in an example in Figure 7.11. Figures 7.11a to 7.11c represent three frames during the initial distribution generation phase. Each frames shows the cells that affect the agents movement (with red, green and yellow borders). During the search phase, the cell initially classified as bridge is re-classified as water, the agent trajectory that was affected by it, is re-initialised to the the point it first interacted with the cell $\mathbf{t}_\tau$ shown in plot 7.11d. plot 7.11e then shows the re-propagation of the agent sample.

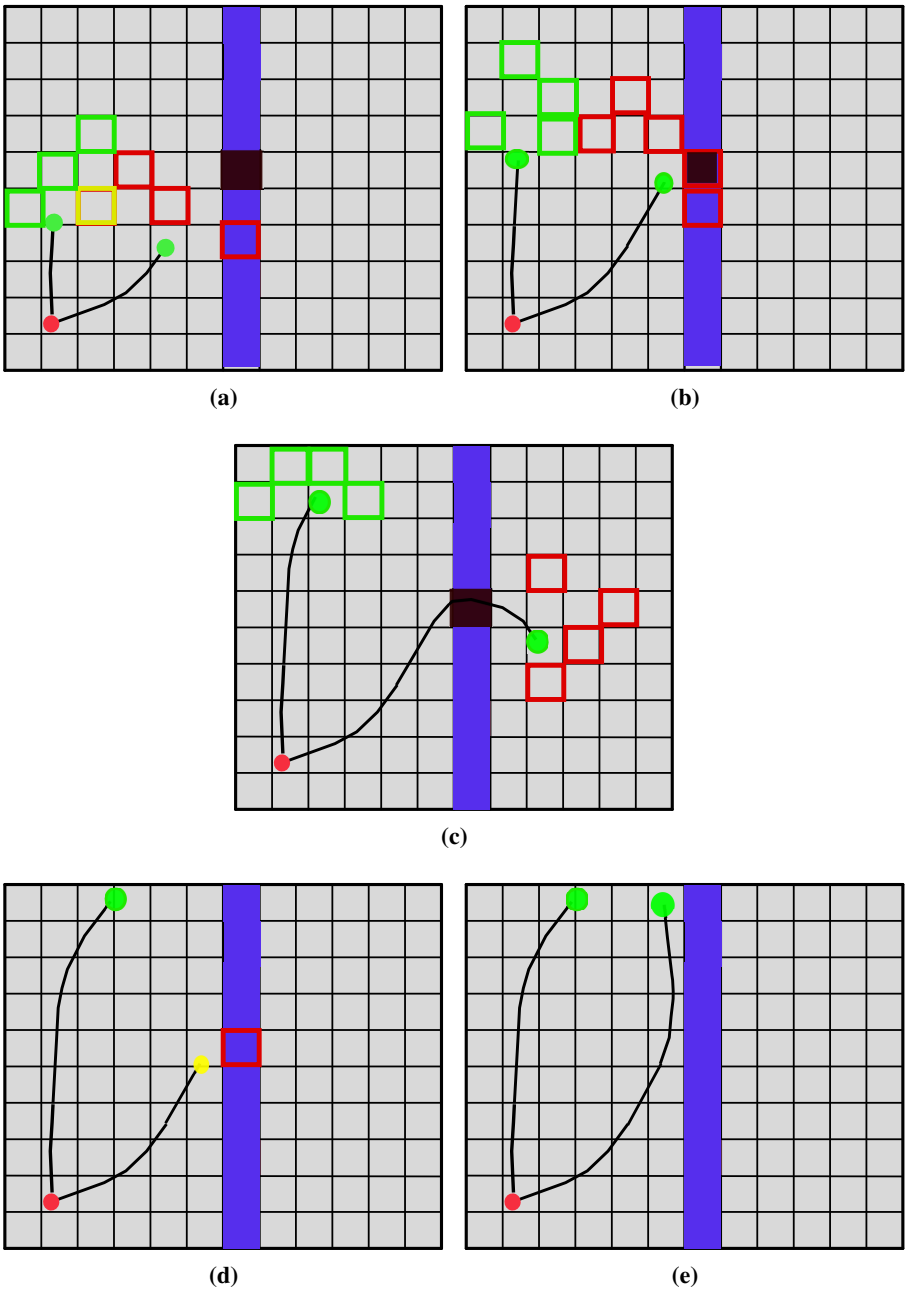
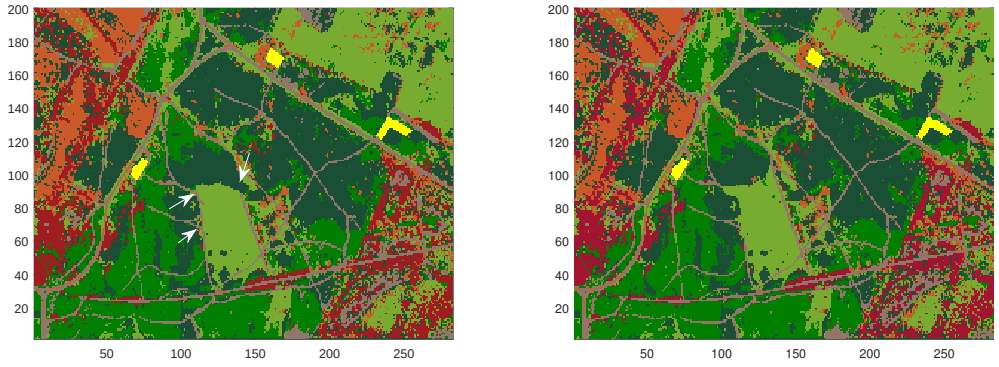


Figure 7.11: Illustration of land cover classification-based update model. The red, green and yellow circles represent $\mathbf{t}_L$. $\mathbf{t}_\tau$ and $\mathbf{t}_S$ respectively. The blue filled cells represent water. The grey cells represent normal ground. The black cell represents bridge over the water. The red , green and yellow squares cell boundaries represent cells that have affected the agent 1, agent 2 and both agent 1 and 2 movement.

.



(a) Initial classification **(b)** Current classification

Figure 7.12: The initial and new land cover classification. Areas that differ to the initial classification are marked by white arrow in Figure 7.12a.

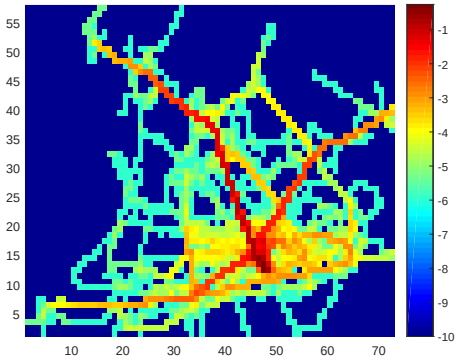


Figure 7.13: The initial distribution generated using the initial representation of search area.

7.6 Evaluation of Land Cover classification Update Model

Since evidence detection can be sparse, this experiment illustrates the importance of considering land cover information in the absence of any evidence information. We do this using one of the Lost Person cases (trail 2) illustrated in Figure 7.4. To perform the experiment, we will use the environment and agent model setup detailed in Section 7.4.

The initial and current land cover classifications are given in Figure 7.12. Parts of the initial representation that differ to current representation are marked by white arrow in Figure 7.12a. The colour scheme is deliberate and used to highlight changes in vegetation and topography. Figure 7.13 illustrates the initial distribution generated using the initial land cover data sets T and V. Consider the case where a Lost Person has traversed the search area as per Figure 7.14.

Similar to experiment performed in Section 7.4, we assume the LP was reported lost in location corresponding to cell red circle in Figure 7.4 and was walking in the direction of arrow. For this simple proof of concept experiment, only 500 agent particles are initiated at PLS and

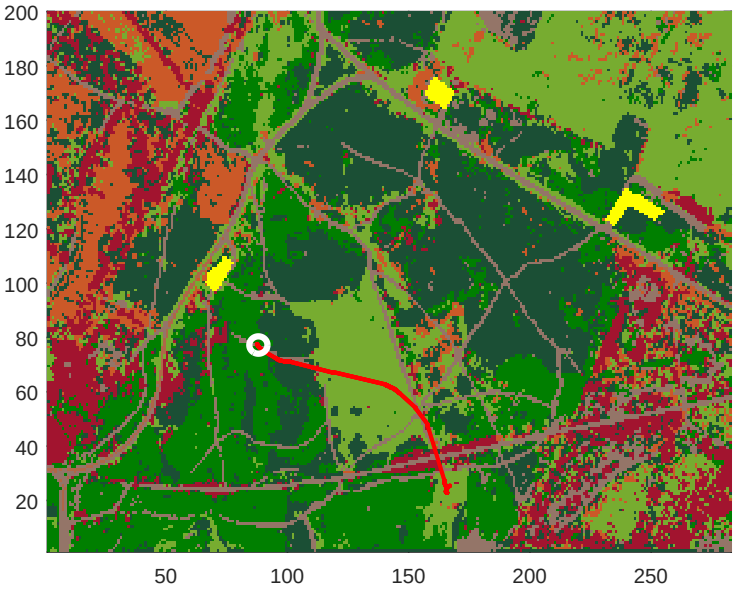


Figure 7.14: Example scenario of Lost Person movement for classification-based update. The white circle represent the location where the person became stationary.

propagated.

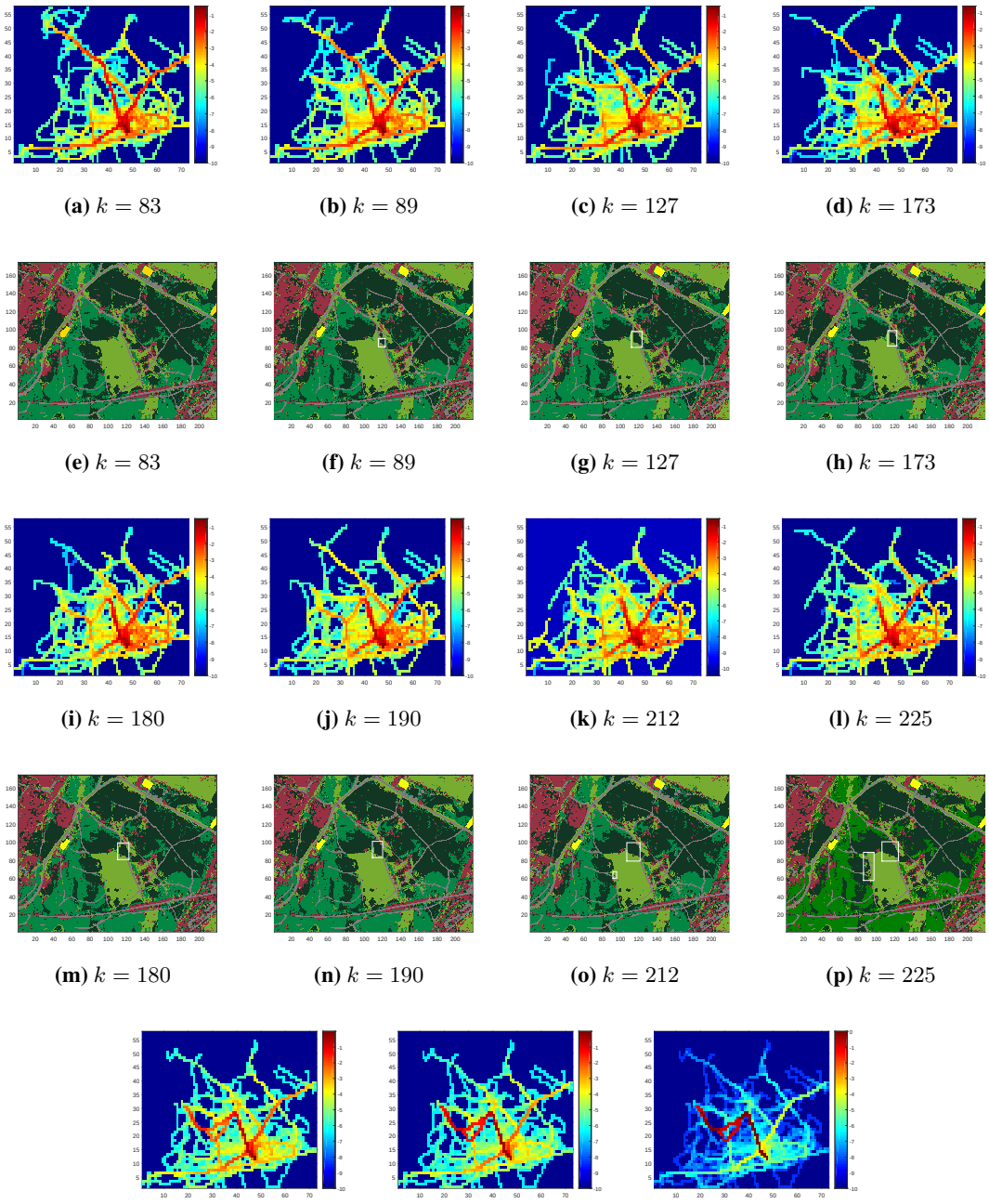
7.6.1 Evaluation Results

As the UAV starts exploring the search area based on the underlying initial distribution, it makes new observations of the land cover. In cases where the re-classification is different to initial classification, trails samples affected are re-propagated using latest representation of the land cover, causing change in the shape of the distribution.

This results in an evolving distribution over LP trail illustrated in Figure 7.15. Looking at the simulation results, it can clearly be seen that considering the updated representation of the land cover has a profound effect on trail representation and helps reduce search times.

In this particular case, the UAV was able to locate the LP quicker by a factor of 1.5 compare to experiment results in Section 7.4.

The importance of considering new classification of the land cover and keeping a history of agent trail (in response to the limitations of diffusion based models discussed in Section 2.7) is further illustrated in Figure 7.16. Plots 7.16a and 7.16b show the initial distribution generated using the agent model. Plots 7.16c to 7.16h show the update of the distribution at different times during the search operation using the updated land cover classification given in Figure 2.15. As we can see, when it is determined that based on updated representation of the land cover, cells initially classified as bridge are re-classified as water, the trail particles crossing the bridge are not sampled in the re-sampling step. All child particles avoid crossing the water, which results



(a) $k = 83$ **(b)** $k = 89$ **(c)** $k = 127$ **(d)** $k = 173$

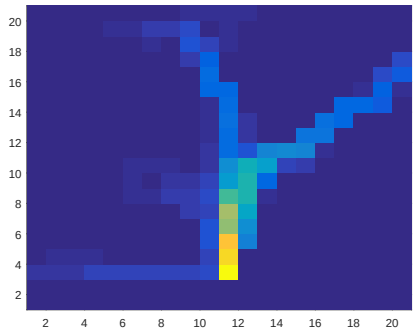
(e) $k = 83$ **(f)** $k = 89$ **(g)** $k = 127$ **(h)** $k = 173$

(i) $k = 180$ **(j)** $k = 190$ **(k)** $k = 212$ **(l)** $k = 225$

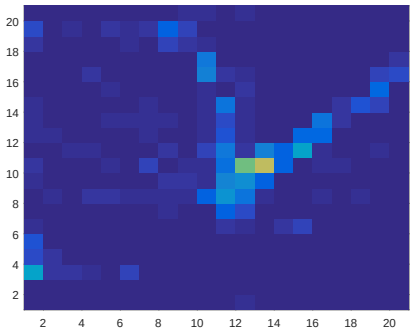
(m) $k = 180$ **(n)** $k = 190$ **(o)** $k = 212$ **(p)** $k = 225$

(q) $k = 255$ **(r)** $k = 257$ **(s)** $k = 259$

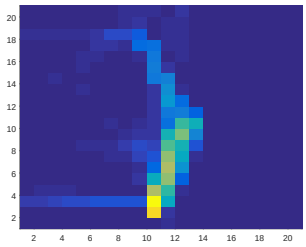
Figure 7.15: Land cover classification-based updated distribution over LP trail at different time-steps given in log.



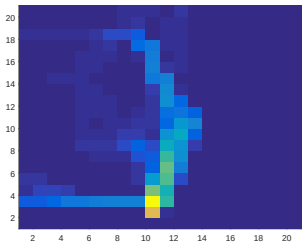
(a) $p(a_i \in \mathbf{T})$ at $k = 1$



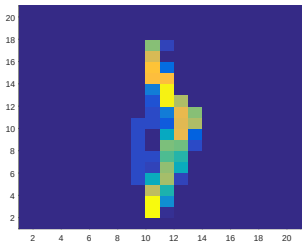
(b) $p(\mathbf{t}_L \in a_i)$ at $k = 1$.



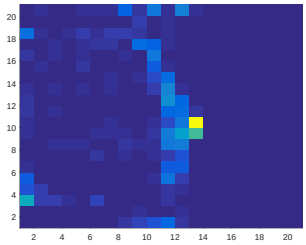
(c) $p(a_i \in \mathbf{T})$ at $k = 8$.



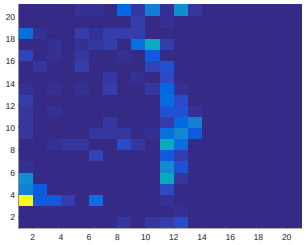
(d) $p(a_i \in \mathbf{T})$ at $k = 20$.



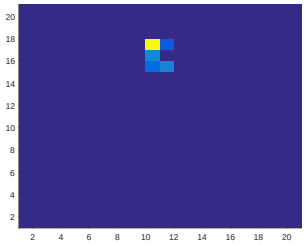
(e) $p(a_i \in \mathbf{T})$ at $k = 54$.



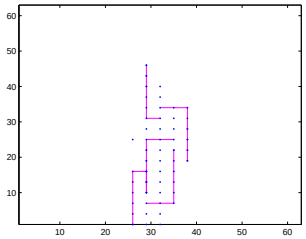
(f) $p(\mathbf{t}_S \in a_i)$ at $k = 8$.



(g) $p(\mathbf{t}_S \in a_i)$ at $k = 20$.



(h) $p(\mathbf{t}_S \in a_i)$ at $k = 54$.



(i) Visited cells.

Figure 7.16: Plots 7.16a and 7.16b presents the generated agent-based initial distribution over both LP trail. Plots 7.16c to 7.16e present the updated distribution over LP trail. Plots 7.16f to 7.16h present the corresponding updated distribution over LP end location. Figure 7.16i shows the cells visited before the LP was detected.

in basically halving the search area and reducing the time-to-locate metric by a factor of 3 compared to grid-based search using diffusion-based prior illustrated Figure 2.15.

7.7 Summary

This chapter detailed the LP trail update model. We considered three types of information: the LP, the evidence and the new land cover, the latter of which results in updated representation of the land cover.

The effect of each information type on inference of LP trail was modelled separately. The performance of the update model was evaluated using two experiments. In the first experiment, it was shown that by considering evidence information in addition to LP, the search times can be reduced by factor of 3 to 25. It was also shown that even in the absence of any evidence, by planning UAV search path using the distribution over the trail of the LP, our search model was able to out perform the conventional grid-based search, especially when grid search was performed using diffusion-based initial distribution.

In the second part of the chapter, we illustrated the importance of considering new land cover information. We showed that even in the absence of evidence, just by considering land cover observations, our search model was able to deal with imperfect initial land cover data sets, adapt to current representation of the land cover and infer the trail taken by the LP, reducing the search time by a factor of up to 3. As a result, we can say that combining and utilising the three information types can significantly improve search times improving survivability of the LP.

So far in this thesis, we have assumed that the platform searching the environment is able to detect both the LP and the evidence deposited by the LP accurately and that there are no uncertainties in the position of either. This simplifying assumption was to develop the basic algorithms. The goal now is to move beyond to look at the constraints on practical platforms. To do this, we need to investigate localisation methods that can localise the searching platforms position and that of the detected evidence.

Chapter 8

Detection and Localisation in Unknown Environment

8.1 Introduction

Having detailed elements of our proposed search model in previous chapters, the purpose of this chapter is to bring together all the elements from the different parts of the thesis into a single coherent system. As explained in the introduction chapter, the full system could not be implemented for logistical reasons. Therefore, this chapter illustrates how the system (the proposed search model as a whole) works using a simulated scenario with real land cover data sets.

We begin by exploring how to extend the work from the previous chapters to allow for mapping from a platform. We discuss feature localisation and why it is required in Section 8.2. Then we give implementation details of the localisation method called SLAM in Section 8.3. Since we use a single observation sensor, which can give us bearing information related to the observed features, we investigate some of the well known bearing only SLAM methods and select one that meets our needs. We present the implementation details in Section 8.4. Then we bring together all the elements of the proposed search framework in simulated experiments and analyse the results in Section 8.5. We show that using our proposed search model, even in the presence of uncertainty in the position of search platform and feature positions, the UAV is able to autonomously search and locate a Lost Person in much shorter time when compared with the grid-based search, which assumes known searching platform and feature positions.

8.2 Feature Localisation

A feature is a distinctive part of the search area such as evidence deposited by the LP, the LP or any other salient feature. Features can be detected by a sensor on board a search platform and can be described with a set of parameters. However, the problem lies in localising the observed

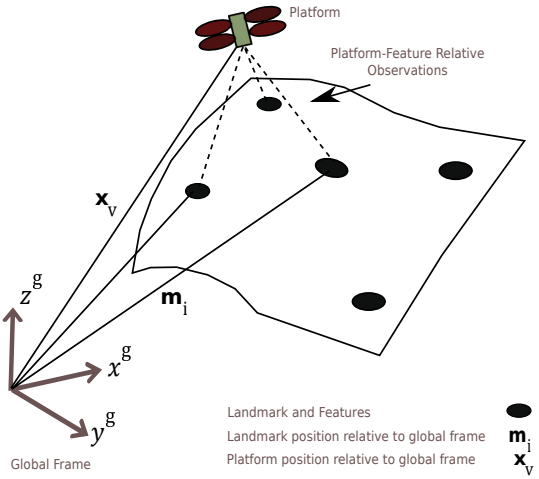


Figure 8.1: Relative observation in SLAM

features with the searching platform – knowing their exact position in the environment. We need to know this for two reasons: First, the search platform must maintain some notion of where it is so that it can inform its controller of its location and allow it to make decisions on where to travel to next or what actions to take. Second, to relate the feature with other information already gathered and update the distribution on the LP whereabouts.

The problem of localising point-like targets or features from a search platform is a heavily studied problem. The techniques that have been proposed and used to achieve this localisation are collectively called *Simultaneous Localisation and Mapping*(SLAM). Starting from the estimation-theoretic foundations of this problem developed in [177], the initial work by Smith et al. [178] and Durrant-Whyte [179] established a statistical basis for describing relationships between features and manipulating the geometric uncertainty.

SLAM assumes an autonomous platform is equipped with a sensor or sensors capable of making measurements of the location of features (salient features, evidence, the LP) relative to the platform. It is assumed that the feature recognition algorithms such as the ones in [180, 181, 181–183] are available to detect these features. The platform starts at a known or unknown location with no knowledge of the location of features in the environment. As the platform moves through the environment, it makes relative observations of the features. Using these observations, it incrementally builds a map of the region explored and uses this to localises the search platform position.

Application of SLAM is wide ranging, encompassing ground-based localisation [184, 185] to aerial localisation [186–189] and to underwater localisation [190–192].

In robotics literature, a number of approaches have been proposed such as Particle Filter based SLAM [193], Grid-based SLAM [194], Graph SLAM [195], Kalman Filter SLAM

[196], Hybrid Metric Map (HyMM) [197] and Simultaneous Localisation and Dense Mapping [198].

A new development in the field of mapping has been the introduction of pose graph SLAM [199]. This approach uses relative pose constraints to build a model of the environment. The main idea behind this methodology is that registering overlapping perceptual data for example optical imagery [200], introduces spatial drifts free edge constraints into the pose graph. These spatial constraints allow the robot to close the loop when revisiting a previously visited place, thereby resetting any accumulated dead reckoning error. This SLAM approach has been proven to produce faster and better results than other SLAM methods. However as the aim in this thesis is to illustrate the importance of localisation methods in autonomous UAV-based search and provide a proof of concept for our proposed search framework, a Kalman Filter based SLAM was used.

8.2.1 Probabilistic Model of Simultaneous Localisation and Mapping

As mentioned in the previous section, the way SLAM works is by augmenting the feature locations observed to a map in some global reference frame while simultaneously using this map to compute the absolute platform location. Therefore, SLAM performs estimation of both the trajectory of the platform and the location of all features real-time without the need for any a initial distribution knowledge of the environment.

Using the notation from [187], the system state $\mathbf{x}_k$ is composed of the platform state $\mathbf{x}_{v,k}$ and the map (feature) states $\mathbf{m}_i$ at time k

$$\mathbf{x}_k = \left[\begin{array}{cccc} \mathbf{x}_v & \mathbf{m}_i & \cdots & \mathbf{m}_N \end{array} \right]_k^T, \tag{8.1}$$

where i is from 1 to N, N being the total number of features initialised and subscript v stands for the platform.

SLAM consists of three main steps, *prediction*, *observation* and *update*. The predict step estimates the future state of both platform and features given a distribution over the current state.

The joint predicted density of the feature locations and platform state at time k given the recorded observations up to time-step $k-1$, $\mathbf{Z}_{k-1}$, and control input up to and including time-step k, $\mathbf{U}_k$ is given by Chapman-Kolmogorov equation as

$$p\left(\mathbf{x}_k, \mathbf{m} | \mathbf{Z}_{k-1}^o, \mathbf{U}_k, \mathbf{x}_B\right) = \int p\left(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{u}_k\right) p\left(\mathbf{x}_{k-1}, \mathbf{m} | \mathbf{Z}_{k-1}^o, \mathbf{U}_{k-1}, \mathbf{x}_B\right) d\mathbf{x}_{k-1}, \tag{8.2}$$

where $p\left(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{U}_k\right)$ is the temporal state transition density also referred as the *process model*. It is conditioned on the estimate of the search platform and map states in the previous step $p\left(\mathbf{x}_{k-1}, \mathbf{m}|\mathbf{Z}^o_{k-1}, \mathbf{U}_{k-1}, \mathbf{x}_B\right)$.

When a new observation is acquired by the sensors on board the platform, it is used to update the state estimate. The Bayesian formulation of SLAM computes the posterior distribution from the posterior of the previous time-step in a recursive manner according to

$$p\left(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}^o_k, \mathbf{U}_k, \mathbf{x}_B\right) = \eta \cdot p\left(\mathbf{z}^o_k|\mathbf{x}_k, \mathbf{m}\right) p\left(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}^o_{k-1}, \mathbf{U}_k, \mathbf{x}_B\right), \tag{8.3}$$

where $p\left(\mathbf{z}^o_k|\mathbf{x}_k, \mathbf{m}\right)$ is the observation model, $p\left(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}^o_{k-1}, \mathbf{U}_k, \mathbf{x}_B\right)$ is the prior distribution or the prediction of probability distribution of the state $\mathbf{x}_k$ and $p\left(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}^o_k, \mathbf{U}_k, \mathbf{x}_B\right)$ is the posterior distribution or the corrected / updated probability distribution of the state [1].

Because the Bayesian representation of SLAM has no closed form solution except when both the state dynamics and observations are linear, we use a Kalman Filter (KF) implementation of SLAM. Only the first two moments of the joint distribution over the platform and the map (mean $\hat{\mathbf{x}}$ and covariance $\mathbf{P}$) are propagated making the process of estimation computationally feasible.

8.2.2 Kalman Filter-Based SLAM

With KF it is assumed that both models are linear. In reality however, this is rarely the case. For example, a robot that moves with constant translational and rotational velocity typically moves on a circular trajectory, which cannot be described by a linear dynamics model. This observation, along with the assumption of uni-modal beliefs, makes plain KF inapplicable to all but the most trivial robotics problems. The Extended Kalman Filter (EKF) overcomes the linearity assumption.

The Extended Kalman Filter (EKF) is an extension of the linear KF [201] with a non-linear process model or non-linear observation model or both, which are linearised around the most-likely state of the system. EKF is used to represent the SLAM posterior using the first two moments (mean $\hat{\mathbf{x}}$ and a covariance matrix $\mathbf{P}$). The mean describes the most likely state of the robot and features, and the covariance matrix keeps the pairwise correlations between all pairs of state variables. The mean and covariance are propagated in time using the first order linearisation of the non-linear system.

Denoting $(k|k-1)$ the estimate at time step k given observations up to and including time step $k-1$, in the stochastic map formulation of the SLAM problem [202, 203], the estimate

[1] For simplicity, we will omit the superscript 'o' from here on.

of $\mathbf{x}_k$ at time k given observations up to time step $k-1$ is $\left\{\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}\right\}$, where $\hat{\mathbf{x}}_{k|k-1}$ is the estimated mean and $\mathbf{P}_{k|k-1}$ the estimated covariance. As in (8.1), the term $\hat{\mathbf{x}}_{k|k-1}$ can be decomposed into the platform and map states,

$$\hat{\mathbf{x}}_{k|k-1} = \begin{bmatrix} \hat{\mathbf{x}}_v \\ \hat{\mathbf{m}}_1 \\ \vdots \\ \hat{\mathbf{m}}_N \end{bmatrix}_{k|k-1},$$

and the uncertainty in the estimated state is represented in a covariance matrix form $\mathbf{P}(k \mid k-1)$ as

$$\mathbf{P}_{k|k-1} = \begin{bmatrix} \mathbf{P}_{vv} & \mathbf{P}_{vm_1} & \cdots & \mathbf{P}_{m_N v} \\ \mathbf{P}_{m_1 v} & \mathbf{P}_{m_1 m_1} & \cdots & \mathbf{P}_{m_1 m_N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{m_N v} & \mathbf{P}_{m_N m_1} & \cdots & \mathbf{P}_{m_N m_N} \end{bmatrix}_{k|k-1}.$$

On diagonals $\mathbf{P}_{vv}$ is the search platform covariance, $\mathbf{P}_{m_1 m_1}$ to $\mathbf{P}_{m_N m_N}$ are the feature covariance values N being the number of features observed and added to the state. The off diagonal matrices are the cross correlations between the platform and map ($\mathbf{P}_{m_1 v}$ for cross correlation between the first feature in the state and the UAV platform) and between the features themselves i.e $\mathbf{P}_{m_1 m_N}$).

The entire structure of SLAM critically depends on maintaining this joint covariance matrix for consistent behaviour and a reliable solution [177, 202]. This is because the estimated locations of the features in the map are not independent of one another and the observations noise from other features observed earlier. Hence the covariance matrix captures the degree of uncertainty between all features and determines how it should deform in response to the observation of features in the map. As the platform moves through the environment taking observations of individual features, the error in the estimates of the relative location between different features reduces monotonically.

The error in the estimate is given as $\tilde{\mathbf{x}}_{k|k-1} = \mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1}$ and the estimate is said to be *covariance consistent* if it obeys the condition

$$\mathbf{P}_{k|k-1} - \mathbf{E}\left[\tilde{\mathbf{x}}_{k|k-1}\tilde{\mathbf{x}}_{k|k-1}^T\right] \geq 0,$$

where ≥ 0 means that the difference is a positive semi-definite.

As mentioned earlier in Section 8.2.1, in the prediction stage, the platform pose is propa-

gated in time according to (8.2). The process model, also called the state transition density, is modelled in a functional form as

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{f}\left(\mathbf{x}_{k-1|k-1}, \mathbf{u}_k, \mathbf{v}_k, \Delta k\right), \tag{8.4}$$

where $\mathbf{f}(\cdot,\cdot,\cdot)$ is the state transition equation at time k producing the current state for the platform and map $\hat{\mathbf{x}}_k$ from previous state $\mathbf{x}_{k-1}$ and $\mathbf{u}_k$ is the control input, $\mathbf{v}_k$ is the process noise vector - a zero mean noise with the noise vector covariance $\mathbf{Q}_k$, and Δk is the length of the time-step $k-1$.

Similarly, the covariance of the state is predicted in time using

$$\mathbf{P}_{k|k-1} = \nabla\mathbf{F}_k\mathbf{P}_{k-1|k-1}\nabla\mathbf{F}_k^T + \nabla\mathbf{G}\mathbf{Q}_k\nabla\mathbf{G}^T. \tag{8.5}$$

Since the dynamics model (8.4) is a non-linear function, it is linearised using Taylor Expansion. This constructs linear approximations taking advantage of the partial derivative of the non-linear function. To do so, the Jacobian of a non linear function is computed.

$\nabla\mathbf{F}$ and $\nabla\mathbf{G}$ are the Jacobians of the state transition function (8.4) with respect to the predicted state vector at previous time step $\hat{\mathbf{x}}_{k-1|k-1}$ and the noise vector $\mathbf{v}_k$ respectively and $\mathbf{P}_{k-1|k-1}$ is the updated covariance matrix at previous time step.

Similar to process model, the functional form of observation mode or likelihood model is

$$\mathbf{z}_k^i = \mathbf{h}\left(\mathbf{x}_k, \mathbf{m}_i, \mathbf{w}_k\right), \tag{8.6}$$

where $\mathbf{h}(\cdot,\cdot,\cdot)$ is the functional form of the observation model, $\mathbf{z}_k^i$ is the observation of the i^{th} feature and $\mathbf{w}_k$ is the zero mean observation noise with associated covariance $\mathbf{R}_k$.

Once an observation happens, and if the feature has already been initialised and has an estimate in the state, it is updated by linearising the observation function (8.6) about the state estimate $\hat{\mathbf{x}}_{k|k-1}$ and then performing the Kalman update [187]

$$\mathbf{x}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{W}_k\boldsymbol{\nu}_k, \tag{8.7}$$

$$\mathbf{P}_{k|k} = \mathbf{L}_k\mathbf{P}_{k|k-1}\mathbf{L}_k^T + \mathbf{W}_k\mathbf{R}_k\mathbf{W}_k^T, \tag{8.8}$$

where

$$\mathbf{L}_k = \mathbf{I} - \mathbf{W}_k\nabla\mathbf{H}_k,$$

$$\mathbf{W}_k = \mathbf{P}_{k|k-1}\nabla\mathbf{H}_k^T\mathbf{S}_k^{-1},$$

$$\boldsymbol{\nu}_k = \mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1},$$

$$\mathbf{S}_k = \nabla\mathbf{H}\mathbf{P}_{k|k-1}\nabla\mathbf{H}^T + \mathbf{R}_k,$$

where $\mathbf{W}_k$ is the Kalman gain, which is iteratively corrected by KF causing the estimation of the state vector to converge towards an optimal solution, $\nabla\mathbf{H}$ is the Jacobian of the non-linear observation function $\mathbf{h}(.,.)$ with respect to the predicted state $\hat{\mathbf{x}}_{k|k-1}$ derived from the non-linear observation model, $\boldsymbol{\nu}_k$ is the innovation (the difference between the actual and the predicted observation) and $\mathbf{S}_k$ is the innovation covariance.

If the observed feature is new, its estimate is given by applying the *inverse observation function*.

$$\mathbf{x}'_{i,k|k} = \mathbf{g}[\mathbf{x}_k, \mathbf{z}_{i,k}, \mathbf{w}_k], \tag{8.9}$$

then augmenting it to system state,

$$\hat{\mathbf{x}}_{k|k} = \begin{bmatrix} \hat{\mathbf{x}} \\ \mathbf{x}'_i \end{bmatrix}_{k|k}, \tag{8.10}$$

and computing the updated covariance,

$$\mathbf{P}_{i,k|k} = \nabla\mathbf{G}_i^x\mathbf{P}_k\nabla\mathbf{G}_i^{xT} + \nabla\mathbf{G}_i^w\mathbf{R}_k\nabla\mathbf{G}_i^{wT}, \tag{8.11}$$

$$\mathbf{P}'_{k|k} = \begin{bmatrix} \mathbf{P} & \mathbf{P}\nabla\mathbf{G}_i^{xT} \\ \mathbf{P}\nabla\mathbf{G}_i^x & \mathbf{P}_i \end{bmatrix}_{k|k}, \tag{8.12}$$

where $\nabla\mathbf{G}_i^x$ and $\nabla\mathbf{G}_i^w$ are the Jacobians of function $\mathbf{g}(.,.)$ with respect to the platform and observation noise respectively [204].

8.3 Simultaneous Localisation and Mapping Implementation

There are different implementations of KF SLAM. The type of implementation depends on two things: (a) the type of platform (aerial, ground-based or under water), (b) the type of sensors used by the platform to capture the environment (monocular vision camera, stereo vision camera, sonar, laser, etc).

In this research, because we deal with aerial platforms that fly using rotors and control parameters such as roll, pitch and yaw, we investigate the SLAM implementation with 6 Degrees

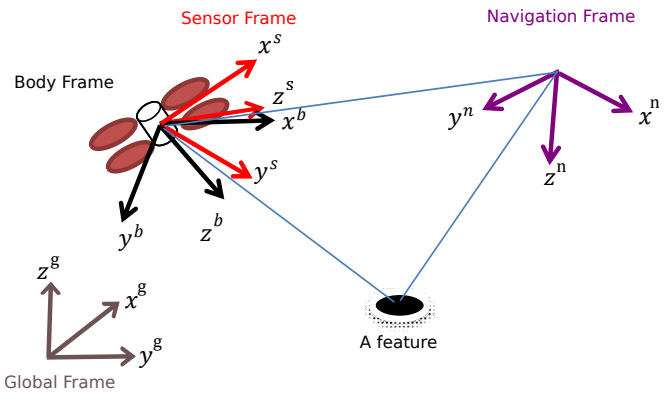


Figure 8.2: Illustration of the four different coordinate frames.

of Freedom (DoF) [2] to address (a).

However, due pay-load limitations on our platforms, mentioned in Section 1.2.4, we can only have one detection sensor capturing the environment. We choose this to be a monocular vision camera since we are interested in detection of environment, evidence deposited and the LP. As a result of this, we will further investigate and select a Monocular SLAM initialisation method to address (b).

Since we have detailed the type of platform and sensors considered for this research in Section 1.2.2, we continue to provide an overview of SLAM implementation for a monocular vision sensor called Monocular SLAM.

However before going into detail of Monocular SLAM and because the Monocular SLAM we are considering is based on strap down INS and other sensors such as vision and GPS, each taking measurements with respect to a coordinate frame, we give a general description of coordinate frame types used.

8.3.1 Coordinate Frames

There are four main types of coordinate frames,

- *Body Coordinate Frame (b):* This coordinate frame consists of a set of three orthogonal axes rigidly attached with its origin at the centre of mass of the platform. For example, in case of the UAV in Figure 8.2, the x-axis (roll) points out to the right side of the platform, y-axis (pitch) points forward and z-axis (yaw) points down .

- *Inertial Navigation Coordinate Frame (n):* This coordinate frame is a non-accelerating frame, the origin of which could be any point in the universe with three mutually orthogonal axes. Inertial sensors measure the acceleration and rotation rate with respect to this

[2] 6 DoF refers to motion of a rigid body in three-dimensional space, namely the ability to move forward/backward, up/down, left/right (translation in three perpendicular axes) combined with rotation about three perpendicular axes (pitch, yaw, roll).

inertial frame.

- *Sensor Coordinate Frame (s):* This frame used for sensors such as vision has its own coordinate frame. The origin of this coordinate frame is the location of the sensor and its axes are defined differently depending on the type of the sensor, each sensor used has a coordinate frame that may be different to others[3].

- *Global Coordinate Frame (g):* The global frame is defined as the local tangent plane to the earth surface at the origin. By definition, the body and global frame coincide when the quad-rotor is at coordinate (0,0,0) and its altitude is zero in all three angles. The global frame is defined as the local tang

8.3.2 6 DoF SLAM Implementation

8.3.2.1 State of the system

For 6 DoF implementation, the platform state $\mathbf{x}_k$ (8.1) is given by

$$\mathbf{x}_{v,k} = [\ \mathbf{p}^n \quad \mathbf{v}^n \quad \mathbf{\Psi}^n\ ]_k^T ,$$

which consists of the platform position $\mathbf{p}_k^n = [x^n\ y^n\ z^n]^T$, velocity $\mathbf{v}_k^n = [u^n\ v^n\ w^n]^T$ and the attitude $\mathbf{\Psi}_k^n$, all expressed in the navigation frame n, where the subscript v stands for vehicle. The physical constraints of the platform mean that the pitch and roll angles cannot exceed 19.4°, avoiding singularities in the system. Therefore, the attitude is parameterised as the standard roll-pitch-yaw Euler parametrisation, $\mathbf{\Psi}_k^n = [\phi^n\ \theta^n\ \psi^n]$. Euler angle represents the attitude of the platform in terms of three successive rotations of the navigation frame to body frame. We use the standard aerospace convention which is roll followed by pitch followed by yaw , as illustrated in Figure 8.3.

The map state $\mathbf{M}$ is a collection of the positions of each feature $\mathbf{M} = \{\mathbf{m}_1^n, \cdots, \mathbf{m}_N^n\}$ where each feature is represented by

$$\mathbf{m}_i^n = [\ x_i^n \quad y_i^n \quad z_i^n\ ]^T . \tag{8.13}$$

Therefore, the overall dimension of the state vector is proportional to the number of features in the map.

[3] In our case for the experiments we have carried out, we assume the axes of the vision sensor is the same as the body frame.

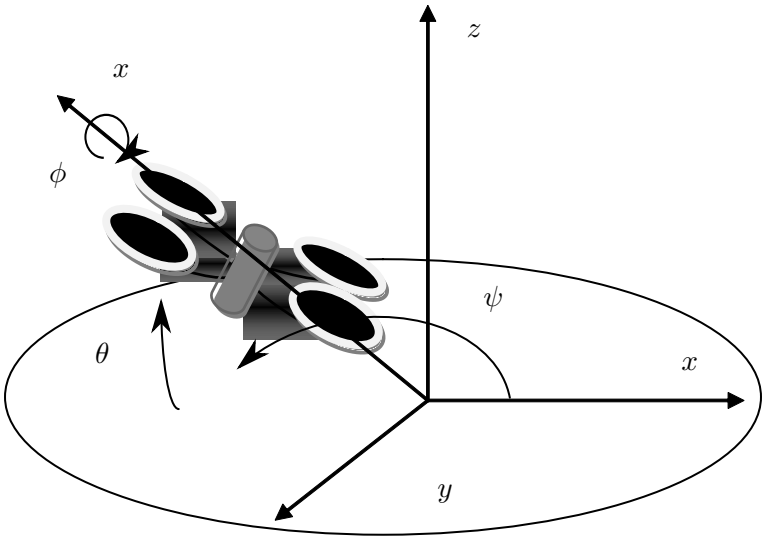


Figure 8.3: Roll Pitch Yaw

8.3.2.2 Process Model

Since the state represents both platform and feature states, the process model in (8.4) consists of the platform and map dynamic model,

$$\begin{bmatrix} \hat{\mathbf{x}}_{v,k} \\ \hat{\mathbf{m}}_{i,k} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_v\left(\mathbf{x}_{v,k-1}, \mathbf{u}_{v,k}, \mathbf{v}_{v,k}, \Delta T\right) \\ \mathbf{f}_{m_i}\left(\mathbf{x}_{m_i,k-1}, \mathbf{u}_{m_i,k}, \mathbf{v}_{m_i,k}, \Delta T\right) \end{bmatrix}, \tag{8.14}$$

where $\mathbf{f}_v$ and $\mathbf{f}_{m_i}$ are platform and feature state transition functions respectively. The control input $\mathbf{u}_{v,k}$ is the linear and angular acceleration measured by an on-board tri-axial IMU, the process noise vector $\mathbf{v}_{v,k}$ is dominated by the measurement errors in the IMU. Assuming features including the evidence deposited by the LP are stationary in the world, $\mathbf{u}_{m_i,k}$ and $\mathbf{v}_{m_i,k}$ are set to zero.

The process noise $\mathbf{v}_{v,k}$ is modelled as a zero mean with associated covariance $\mathbf{Q}_k$ modelling the un-modelled rates between the IMU samples.

UAV and Feature Dynamics Models: The process model $\mathbf{f}_v\left(\mathbf{x}_{v,k-1}, \mathbf{u}_{v,k}, \mathbf{v}_{v,k}, \Delta T\right)$ is a strap down INS algorithm, computing the position, velocity and attitude of the aerial platform from the inertial measurement inputs in the earth- fixed local tangent frame given as [187]

$$\begin{bmatrix} \mathbf{p}_k^n \\ \mathbf{v}_k^n \\ \mathbf{\Psi}_k^n \end{bmatrix} = \begin{bmatrix} \mathbf{p}_{k-1}^n + \mathbf{v}_k^n \Delta k \\ \mathbf{v}_{k-1}^n + \left\{ \mathbf{C}_{b,k-1}^n [f_k^b + \mathbf{v}_{v,k} + \boldsymbol{\delta} f_k^b] + \omega_k^b \right\} \Delta k \\ \mathbf{\Psi}_{k-1}^n + \mathbf{E}_{b,k-1}^n [\omega_k^b + \mathbf{v}_{v,k} + \boldsymbol{\delta} \omega_k^b] \Delta k \end{bmatrix}, \tag{8.15}$$

where f_k^b and ω_k^b are the acceleration and rotation rates measured in the body frame with $\boldsymbol{\delta} f_k^b$ and $\boldsymbol{\delta}\omega_k^b$ representing the noises of both the acceleration and rotation rates respectively. $\mathbf{C}_{b,}^{n}$ is a direction cosine matrix and $\mathbf{E}_{b,}^{n}$ is the transformation matrix that transforms the rotation rate in the body frame to the Euler angle in navigation frame. Details of computing $\mathbf{C}_{b,}^{n}$ and $\mathbf{E}_{b,}^{n}$ and their inverse $\mathbf{C}_{n,}^{b}$ and $\mathbf{E}_{n,}^{b}$ are given in Appendix F.2.1.

Since features are station, their process model is trivially

$$\mathbf{m}_{i,k} = \mathbf{m}_{i,k-1}.$$

8.3.2.3 Observation Model

In our case, there are two classes of observation model: GPS and camera observations.

For the GPS, the observation is a linear model and is a function of the platform state only,

$$\mathbf{z}_{gps,k} = \mathbf{H}_{gps,k}\mathbf{x}_k + \mathbf{w}_{gps,k}.$$

This observation model estimates the GPS observations $\mathbf{z}_{gps,k}$ using the observation matrix $\mathbf{H}_{gps,k}$, which relates the output of the sensor to the state vector $\mathbf{x}_k$. $\mathbf{w}_{gps,k}$ is a temporally uncorrelated observation errors with zero mean and covariance $\mathbf{R}_{gps,k}$ such that

$$\mathbf{E}\left[\mathbf{w}_{gps,k}\right] = 0,$$

$$\mathbf{E}\left[\mathbf{w}_{gps,k}\ \ \mathbf{w}_{gps,k}^T\right] = \mathbf{R}_{gps,k} = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_z^2 \end{bmatrix}_k.$$

For the i^{th} feature, an on-board vision camera makes relative bearing observations $\mathbf{z}_{i,k}$ to features within the sensor frame. This non-linear observation model relates the platform's relative observations of the features (using the on-board vision sensor) to the state given in (8.6) by computing

$$\mathbf{z}_{i,k} = \mathbf{h}\left(\mathbf{p}_k^n, \boldsymbol{\Psi}_k^n, \mathbf{m}_{i,k}^n, \mathbf{w}_k\right), \tag{8.16}$$

where $\mathbf{z}_{i,k}$ is the observation of the i^{th} feature (expressed in pixel coordinates) and $\mathbf{w}_k$ is the observation error. The noise model and the error covariance are detailed later in this section.

Computing Vision Sensor Observation: Since we are using a vision camera to capture the environment, the observation is better represented as pixels in the image of the camera, using a

pinhole camera model,

$$\mathbf{z}_{i,pix,k} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_u \left(\frac{y^s}{x^s} + u_0\right) \\ f_v \left(\frac{z^s}{x^s} + v_0\right) \end{bmatrix}_k , \tag{8.17}$$

where x^s, y^s and z^s are the position of the feature in the sensor coordinate frame as defined in Section 8.3.1 and u_0, v_0, f_u, and f_v are calibration parameters for the camera with pixel noise covariance $\mathbf{R}_{pix}$.

Having detailed the state, process and observation models required by our implementation of SLAM (6 DoF SLAM), we need to find out how new features can be added to the SLAM generated map. This relates to Monocular SLAM initialisation methods.

8.3.3 Monocular SLAM Initialisation Methods

With Monocular SLAM also called bearing only SLAM proposed in [205, 206], when a feature is first observed, its depth cannot be estimated straight away. This is because the depth of the feature is not fully observable from a single measurement. As a result, its initial position is poorly known. However, upon successive observations of the feature, the estimate of the feature position including depth can be improved. Therefore, qualitatively, a feature can be considered to be in one of two conditions: *poorly-localised*, and *well-localised*. In the poorly localised condition, the position uncertainty is sufficiently large that a KF representation of Cartesian position performs poorly. In the well-localised condition, the feature uncertainty is sufficiently small that regular KF updates can be applied.

Different approaches have been proposed to tackle this problem of getting features well-localised. Some approaches perform *delayed* and others *undelayed* initialisation of the feature position in the map.

The delayed initialisation assumes that a feature cannot be used when it is poorly localised. Instead, the initialisation of the feature is delayed until it can be directly initialised into the well-localised condition. This is achieved by waiting until the feature has been observed from a sufficiently large base line α that its position can be estimated accurately using triangulation [187, 196, 207]. The typical value used for α, which is derived experimentally in [186] is 40°. This crucial property of this algorithm makes this method computationally very expensive, requiring large state and covariance updates and many check calculations.

To deal with both the computational cost of the method, and delayed utilisation of feature information into the filter, authors have been spurred to develop undelayed initialisation algorithms that can exploit a feature as soon as it is first observed.

According to [208], the filter can perform more robustly by not having to delay the initialisation. This is because having the bearing information of the features in the map as they are observed permits there immediate use as angular references. In addition, it allows the system to use features that lie close to the direction of motion of the UAV for which baseline would take too long to obtain. This form of initialisation is called undelayed initialisation. To achieve this, many approaches have been proposed.

In [209], a multi hypothesis filtering approach is used to estimated a feature's position along the line of sight when the feature is first observed. Each of the hypotheses about the features position is treated as a separate feature and integrated into the filter. Over time with new observations of the feature from different poses only one of the hypothesis will remain and the rest are removed.

Another undelayed initialisation method proposed in [210] performs bearing only SLAM using Gaussian Sum filter (GSF) to initialise features. Because the initial Probability Distribution Function (PDF) of a feature in the environment cannot be represented using a single Gaussian, the GSF is used for estimation as it can deal with arbitrary PDFs represented as sets of Gaussian distributions. When a feature is observed, a bank of EKFs are initialised for the feature with different hypotheses of its ranges, all using the same sensor with independent operations. Performance of EKFs are evaluated from the likelihood values derived from measurement residuals or innovations. To maintain the number of EKFs, a Sequential Probability Ratio Test (SPRT) is carried out to decide whether to keep or remove the EKFs from the bank. This process is computationally very expensive as with each observations of features, banks of EKFs are generate and the decision procedure is applied. Therefore, with this method the number of hypotheses grow exponentially.

Similarly, in [211, 212] another very popular and well established method is proposed for undelayed initialisation using Unified Inverse Depth Parametrisation (UIDP). This method improves the linearity of the measurement equation even for small changes in the camera position yielding small changes in the parallax angle. This fact allows a Gaussian distribution to cover uncertainty in depth which spans a depth range from nearby to infinity. In this approach features are initialised in the first frame observed with an initial fixed depth and uncertainty determined heuristically to cover ranges from nearby to infinity. Because of its scalability, far features provide very useful information about the platform orientation, reducing the angular drift during long motions. These characteristics make this method very desirable for Monocular SLAM implementation in unstructured environments [213].

This method implies a dimension 6 state vector per feature, doubles the size of the map

state vector. But, the uncertainties in close feature locations collapse after several frames to accurate Gaussian distributions in Euclidean 3D space allowing for safe conversion back to the Cartesian (XYZ) coordinate system. This means that, the long term computational cost of running this method would not increase [211].

However, this method too suffers from two issues: (1) negative depths. This happens when the estimate of the inverse of the depth becomes negative during the KF update, which can cause catastrophic failure in the map estimate. In other words, the depth estimate of a feature can approach infinity one step and be negative the next. (2) it initialises features in the first frame, which means weak image features could be added to the map introducing biases to the system, making it difficult for later matching.

To deal with first issue, a method is proposed in [214] where depth ρ is parameterised by d^{-1} called the Inverse Depth Parameterisation (IDP). It can represent a range from close by to infinity.

To deal with the second issue, other methods such as the one proposed in [215] suggests a delayed IDP method where the inverse depth and issues with new features are dealt with before being added to the state. Although this method is expensive, it is shown that it produces better results. Similarly Saleh et all in [213] suggests the use of inertial systems with IDP to constrain the uncertainties in the prediction of the camera motion between observations reducing linearisation errors on observations. On the other hand Sola in [216] claims both the issue of negative depth and the non-linear observation model of IDP could be dealt with by using anchors. This method is called Anchored Homogeneous Point (AHP) parametrisation.

To asses the performance of both the *delayed* and *undelayed* initialisation methods, we have investigated and implemented the three very well established Bearing only SLAM methods (DI, IDP and AHP). All three utilise the basic SLAM structure, but the way in which they handle the poorly-localised condition is different. Details of the IDP and AHP initialisation along with comparison results with DI can be found in Appendix F.

Based on the results achieved (presented in Appendix F.5.5), although a bit expensive, DI outperforms the other methods. For this reason, we will use and present the Delayed Initialisation (DI) initialisation method next.

To deal with the computational cost of DI, we propose a change based on our findings presented in Appendix F.5.5.1. We modify the augmentation logic so that the state is augmented and the observation is stored only if there is a minimal angle threshold of 28° between successive observations. This brings the computational complexity of DI down closer to the other two methods.

8.4 Delayed Initialisation

In DI we are dealing with multiple views of a feature. Because of this, a three step process is followed, which includes: *storing feature observations*, *initialising the feature*, and *utilising the stored observations*.

Suppose a new feature is first observed at time step k_1 but the baseline is not sufficiently large to initialise the feature until time step k_2. Rather than discard all the feature observations $\mathbf{z}_i(k)$: $k_1 < k < k_2$, they are stored for future processing.

Once it is determined that a sufficient baseline exists between two observations of the feature. the features Cartesian position in the world is calculated by computing the closest point between these two observations in 3D space and added to the state, also updating the covariance. To improve the estimate of both the initialised feature and the platform pose, a batch update is then performed. All the stored observations and the poses corresponding to the feature are used. The final step is to remove all the poses and observations that are not required. From here on when the initialised feature is observed again, it is simply updated using normal EKF steps.

8.4.1 Delayed Initialisation Implementation

8.4.1.1 Storing Feature Observation with Associated Platform (UAV) Pose

When an uninitialised feature is observed, its current bearing observation is stored by augmenting the state and covariance matrix with current platform pose (its position and Euler angle states). Suppose,

$$\hat{\mathbf{x}}^n_{v,k|k-1} = \begin{bmatrix} \mathbf{p}^n \\ \hat{\mathbf{v}}^n \\ \hat{\mathbf{\Psi}}^n \end{bmatrix}_{k|k-1}, \hat{\mathbf{x}}^n_{p,k|k-1} = \begin{bmatrix} \mathbf{p}^n \\ \hat{\mathbf{\Psi}}^n \end{bmatrix}_{k|k-1},$$

where $\hat{\mathbf{x}}^n_{v,k|k-1}$ is the platform's state estimate. This comprises the platform's estimated position $\mathbf{p}^n_k$, velocity $\hat{\mathbf{v}}^n_k$ and attitude $\hat{\mathbf{\Psi}}^n_k$. The $\hat{\mathbf{x}}^n_{p,k|k-1}$ is the platform's pose state estimate comprised of the platform's position $\mathbf{p}^n_k$ and attitude $\hat{\mathbf{\Psi}}^n_k$ at the time of observation k,

$$\hat{\mathbf{x}}^n_{aug,k|k-1} = \begin{bmatrix} \hat{\mathbf{x}}^n_v \\ \mathbf{m}^n_i \\ \hat{\mathbf{x}}^n_p \end{bmatrix}_{k,k-1}, \tag{8.18}$$

where $\hat{\mathbf{x}}^n_{aug,k|k-1}$ is the augmented state vector comprising the platform's states $\hat{\mathbf{x}}^n_v$, the 3D position of all the features in the navigation frame $\mathbf{m}^n_i$ and the added platform pose states $\hat{\mathbf{x}}^n_p$.

Similarly, the covariance augmentation a matrix is given by

$$\mathbf{P}_{aug,k|k} = \begin{bmatrix} \mathbf{P}_{vv} & \mathbf{P}_{vm} & \mathbf{P}_{vp} \\ \mathbf{P}_{mv} & \mathbf{P}_{mm} & \mathbf{P}_{mp} \\ \mathbf{P}_{pv} & \mathbf{P}_{pm} & \mathbf{P}_{pp} \end{bmatrix}_{k|k}, \tag{8.19}$$

$\mathbf{P}_{aug,k|k}$ is the augmented covariance matrix. $\mathbf{P}_{pp}$ is the covariance of the pose states, which is derived by taking the sub-block of the position and attitude covariance matrix from within $\mathbf{P}_{vv}$. These states have the same value and the same covariance as the current platform position and attitude. $\mathbf{P}_{pm}$ is the covariance of the pose state and the map states. It is taken from existing cross correlations between the current platform states and map states[4] and the current pose states.

8.4.1.2 Observation Bearing Computation

Because we are dealing with monocular or bearing only systems, we need to compute the bearing observation of features (azimuth φ_i and elevation υ_i) from the pixel measurement in 8.17. This is done using

$$\mathbf{z}_{i,ang,k} = \begin{bmatrix} \varphi_i \\ \upsilon_i \end{bmatrix} = \begin{bmatrix} \text{atan}\left(\frac{u_i - u_0}{f_u}\right) \\ \text{atan}\left(\frac{v_i - v_0}{f_v}\cos\varphi_i\right) \end{bmatrix}_k, \tag{8.20}$$

The pixel noise is assumed to be additive and Gaussian, therefore the angular noise covariance can be approximated by

$$\mathbf{R}_{ang,k} = \mathbf{R}_{pix,k} \begin{bmatrix} \left(\frac{1}{f_u^2}\right) & 0 \\ 0 & \left(\frac{1}{f_v^2}\cos\varphi_i\right) \end{bmatrix},$$

8.4.1.3 Initialising 3D Feature Position Estimate

Each bearing only observation is represented by a 3D point in the space $\mathbf{y}_i^n$ from where the feature was observed along with a unit vector $\overline{\mathbf{u}}_i^n$ pointing along the line of sight of the observation. For example, for first observation of a feature at time k_1, the point in the sky is computed by

$$\mathbf{y}_{k_1}^n = \mathbf{p}_{k_1}^n + \mathbf{C}_{b,k_1}^n \mathbf{p}_{sb}^b,$$

where $\mathbf{C}_{b,k_1}^n$ is the navigation-to-body frame transformation matrix and $\mathbf{p}_{sb}^b$ is the sensor's offset from the platform's body. Both $\mathbf{p}^n$ and $\mathbf{C}_b^n$ are determined from the stored pose data associated

[4] Sub-block $\mathbf{P}_{vm}$ corresponds only to position and attitude. $\mathbf{P}_{pv}$ is the cross correlation between the current platform state position, velocity and attitude

the observation $\mathbf{p}_{k_1}^n$.

Similarly, the unit vector pointing along the line of sight of the feature observation at time k_1 is computed by

$$\overline{\mathbf{u}}_{i,k_1}^n = \mathbf{C}_{b,k_1}^n \mathbf{C}_s^b \overline{\mathbf{u}}_{i,k_1}^s,$$

where $\mathbf{C}_s^b$ is the transformation between the body and camera frames encoding the nadir angle and $\overline{\mathbf{u}}_{i,k_1}^s$ is the unit vector from the camera is centre to the feature, expressed in sensor-fixed co-ordinates. $\overline{\mathbf{u}}_{i,k_1}^s$ is computed from $\mathbf{z}_{i,ang,k_1}$ and knowledge of the camera's intrinsic parameters by

$$\overline{\mathbf{u}}_{i,k_1}^s = \begin{bmatrix} \cos(\varphi_{k_1})\cos(\upsilon_{k_1}) \\ \sin(\varphi_{k_1})\cos(\upsilon_{k_1}) \\ \sin(\upsilon_{k_1}) \end{bmatrix},$$

Thus, the epipolar line at k_1 starts at $\mathbf{y}_{k_1}^n$ and its direction is given by the unit vector $\overline{\mathbf{u}}_{i,k_1}^n$.

Having computed $\overline{\mathbf{u}}_{i,k_1}^n$ for a feature when it is first observed, every subsequent time the feature is re-observed, $\overline{\mathbf{u}}_{i,k_2}^n$ is computed and it is determined if two observations meet the baseline angle requirement α. This is done by computing

$$\alpha_{k_1:k_2} = \text{acos}\left(|\overline{\mathbf{u}}_{k_1}^n|.|\overline{\mathbf{u}}_{k_2}^n|\right),$$

and checking if the computed angle $\alpha_{k_1:k_2}$ is larger than the baseline angle α. Once it is determined that the two optical rays for the feature meet the baseline angle requirement, the position of the feature is initialised. The rays from the two observations $\overline{\mathbf{u}}_{i,k_1}^n$ and $\overline{\mathbf{u}}_{i,k_2}^n$ should intersect at only one point in 3D space corresponding to the position of the feature $\mathbf{m}_i^n$ illustrated in Figure 8.4.

The feature's position is computed by

$$\mathbf{m}_i^n = \mathbf{g}\left(\mathbf{p}_{k_2}^n, \mathbf{p}_{k_1}^n, \boldsymbol{\psi}_{k_2}^n, \boldsymbol{\psi}_{k_1}^n \mathbf{z}_{k_2}, \mathbf{z}_{k_1}\right), \tag{8.21}$$

where $\mathbf{g}(.)$ is initialisation function. However because the observations and stored platform pose information are noisy, the lines of sight may not intersect perfect, so the initial position of feature is calculated at the closest point between the two lines for each observations.

$$\mathbf{G}\left(\mathbf{p}_{k_2}^n, \mathbf{p}_{k_1}^n, \boldsymbol{\psi}_{k_2}^n, \boldsymbol{\psi}_{k_1}^n \mathbf{z}_{k_2}, \mathbf{z}_{k_1}\right) = \frac{1}{2}\left(\mathbf{y}_{k_2}^n + \mathbf{y}_{k_1}^n + p_1 \overline{\mathbf{u}}_{k_2}^n + p_2 \overline{\mathbf{u}}_{k_1}^n\right),$$

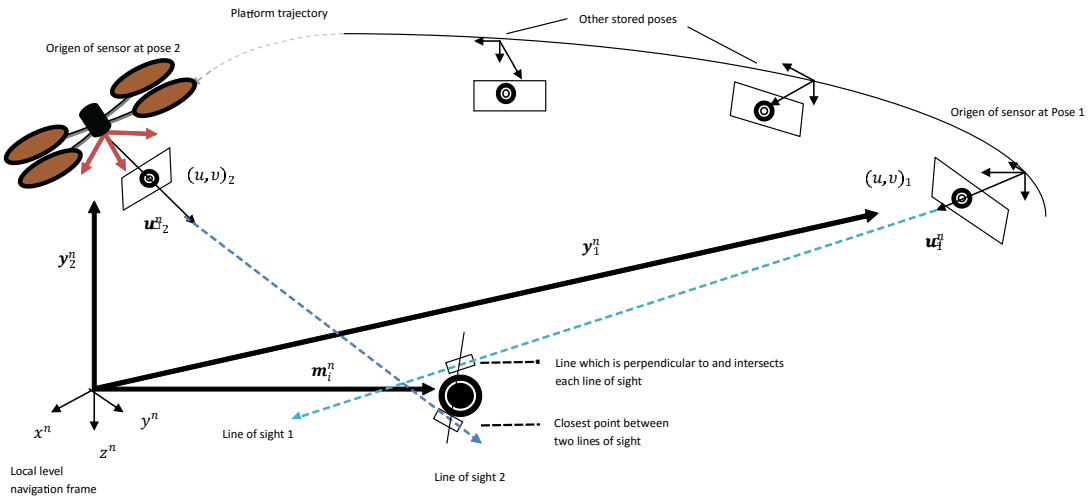


Figure 8.4: Illustration of the delayed initialisation. As could be observed two observations of the same feature with sufficiently large baseline is required to estimate the depth of the feature. The blue rays represent bearings to the detected feature from the two different poses.

where

$$p_1 = \frac{\left[\left(\mathbf{y}^n_{k_2} - \mathbf{y}^n_{k_1}\right) \times \overline{\mathbf{u}}^n_{i,k_1}\right] . \left(\overline{\mathbf{u}}^n_{i,k_1} \times \overline{\mathbf{u}}^n_{i,k_2}\right)}{\mid \overline{\mathbf{u}}^n_{i,k_1} \times \overline{\mathbf{u}}^n_{i,k_2} \mid^2},$$

$$p_2 = \frac{\left[\left(\mathbf{y}^n_{k_1} - \mathbf{y}^n_{k_2}\right) \times \overline{\mathbf{u}}^n_{i,k_1}\right] . \left(\overline{\mathbf{u}}^n_{i,k_2} \times \overline{\mathbf{u}}^n_{i,k_2}\right)}{\mid \overline{\mathbf{u}}^n_{i,k_2} \times \overline{\mathbf{u}}^n_{i,k_1} \mid^2}.$$

Once we have the Cartesian position of the feature, the SLAM filter state and covariance matrix are augmented according to the equations

$$\hat{\mathbf{x}}_{aug,k|k} = \begin{bmatrix} \hat{\mathbf{x}} \\ \mathbf{m}^n_i \end{bmatrix}_{k|k-1}, \tag{8.22}$$

$$\mathbf{P}_{aug} = \begin{bmatrix} \mathbf{I} & 0 \\ \nabla\mathbf{G}_p & \nabla\mathbf{G}_z \end{bmatrix} \begin{bmatrix} \mathbf{P}\,(k) & 0 \\ 0 & \mathbf{R}_{2x2} \end{bmatrix} \begin{bmatrix} \mathbf{I} & 0 \\ \nabla\mathbf{G}_p & \nabla\mathbf{G}_z \end{bmatrix}^T, \tag{8.23}$$

$$\mathbf{R}_{2\times 2} = \begin{bmatrix} \mathbf{R} & 0 \\ 0 & \mathbf{R} \end{bmatrix},$$

where $\nabla\mathbf{G}_p$ and $\nabla\mathbf{G}_z$ are the Jacobians of the initialisation function $\mathbf{g}\,(...)$ w.r.t the pose states $\left(\mathbf{p}^n_{k_1}, \mathbf{p}^n_{k_2}, \boldsymbol{\psi}^n_{k_1}, \boldsymbol{\psi}^n_{k_2}\right)$ and the observations $(\mathbf{z}_{k_1}, \mathbf{z}_{k_2})$ respectively.

8.4.1.4 Incorporating the Information from Remaining Stored Observations

After the feature is initialised, using two observations of it with a good baseline, the remaining stored observations of the feature $\mathbf{z}_{i,k}$, $k_1 < k < k_2$ are incorporated through a single, large

update using the conventional update equations. This update corrects the the pose of the current feature initialised, the platform (UAV) state and all the poses of the other features already in the map. The update uses (8.2.2) to (8.8) for both state and covariance respectively. The innovation or residual is calculated using all the stored observations of the feature.

$$\boldsymbol{\nu}_k = \begin{bmatrix} \mathbf{z}_{i,k_1} - \mathbf{H}\left(\mathbf{p}^n_{k_1}, \boldsymbol{\psi}^n_{k_1}, \mathbf{m}^n_{k_1}\right) \\ \mathbf{z}_{i,k_2} - \mathbf{H}\left(\mathbf{p}^n_{k_2}, \boldsymbol{\psi}^n_{k_2}, \mathbf{m}^n_{k_2}\right) \\ \vdots \\ \mathbf{z}_{i,k_j} - \mathbf{H}\left(\mathbf{p}^n_{k_j}, \boldsymbol{\psi}^n_{k_j}, \mathbf{m}^n_{k_j}\right) \end{bmatrix}, \tag{8.24}$$

where $\nabla\mathbf{H}(k)$ is the composition of Jacobians of the observation function for each stored observation w.r.t the predicted state vector $\hat{\mathbf{x}}_{k|k-1}$.

After the update step has been completed, pose states that no longer have any associated stored observations are removed from the state vector and their corresponding rows and columns are removed from the covariance matrix.

8.4.1.5 Summary of DI SLAM

Having defined the states, process models and observation models, in the *predict* stage, the system state $\hat{\mathbf{x}}_{k-1|k-1}$ and covariance $\mathbf{P}_{k-1|k-1}$ are moved forward in time. The system state is moved forward given the estimated state of the system at the previous time-step $\mathbf{x}_{k-1|k-1}$, using the platform and feature process models in (8.14). Similarly, the system covariance is moved forward given the covariance estimate in previous time-step $\mathbf{P}_{k-1|k-1}$ using the predict step (8.5).

If a new GPS observation is available, this is used to update the system state before the vision sensor based observation is used. The GPS readings are integrated into the 6DoF SLAM using a linear Kalman Filter in the form of (8.7) and (8.8), where the $\mathbf{H}_k$ is simply a state transition matrix that relates the platform state to the observation.

As new observations of a feature arrive, they are converted in to bearing information by applying (8.17) and (8.20) in order. Then one of following steps takes place:

- If the observed feature has is already been initialised in the map, this new observation is used to update and correct the state vector and the covariance matrix using (8.7) and (8.8).

- If the observed feature is not already initialised in the map, and since two observations with baseline angle larger than α are required for a feature's initialisation, both the state and covariance are augmented using (8.18) and (8.19) respectively.

- If the observed feature is not already initialised in the map, but exists in the augmented system state and meets the baseline requirement, it's position is computed from the first and last observation using (8.21). It is then added to the state and covariance using (8.22) and (8.23) respectively. The remaining observations of the feature are used to improve the estimation of the feature's state in a batch update using (8.24) and then removed.

This process of prediction, observation and update is recursively performed to keep localising the platform and the features observed. This results in a well localised map of the explored environment.

The Jacobians for the predict and update of the state is given in Appendix F, Section F.1.

8.4.1.6 Example of 6 Dof SLAM with Delayed Initialisation

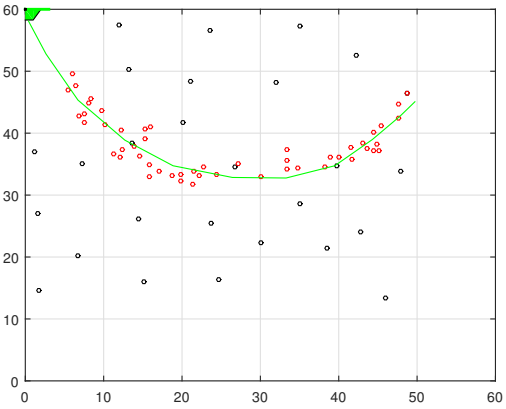
The results of three different 6 DoF Monocular SLAM initialisation implementations (DI, AHP and IDP), along with detailed analysis of the results can be found in Appendix F. Here we show how the DI implementation operates using two example environments containing synthetic features positioned randomly as illustrated in Figure 8.5. The aerial platform is an implementation of the quad-rotor simulation QRSim [29]. It explores the environment at an altitude of 20 meters using a predefined trajectory, which is represented by the green line.

In this experiment we assume that the GPS operates at a rate of 10Hz during the prediction stage, which happens at a faster rate of 25Hz. The IMU sensor run at 30Hz. This mimics the real world use of these sensors where the IMU readings are acquired at a very fast rate to enable the filter to predict the state reliably and the GPS observations happen at much lower rate depending on the environmental conditions.

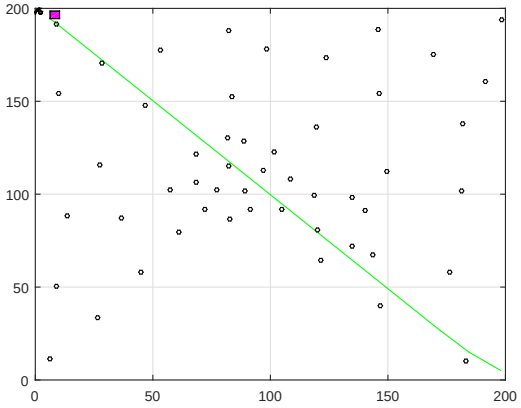
Figure 8.6 and Figure 8.7 show the generated maps. As can be seen in Plot 8.6a, when the features are first initialised, there is some uncertainty associated with feature positions. This is illustrated with ellipses around the feature in case of 2D plots (Plot 8.6a) and spheres in case of 3D plots (plot 8.7d). The larger the ellipse or sphere, the higher the uncertainty of the location of the feature. However, as the platform moves further, and with successive observation of the initialised features, the estimate of the feature position is improved. The uncertainty in the feature's position reduces resulting in well-localised features. This can be seen for 2D maps in plots 8.6b and 8.6c, and in plots 8.7c to 8.7f in case of 3D map.

From these results, it can be seen that the DI implementation of 6 DoF SLAM algorithm produces very consistent performance in all scenarios.

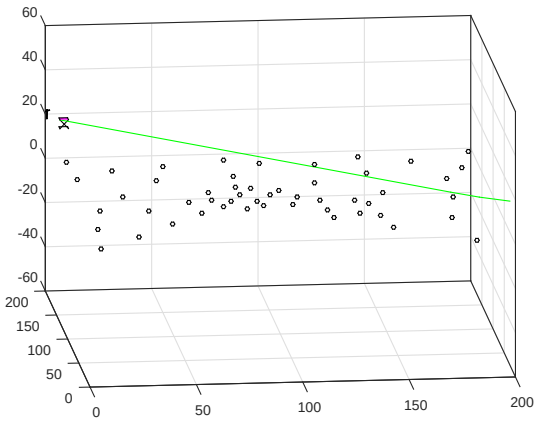
Now having modelled and presented all elements of our proposed search model, we need to show how the complete search model performs.



(a) First experiment scenario 2D

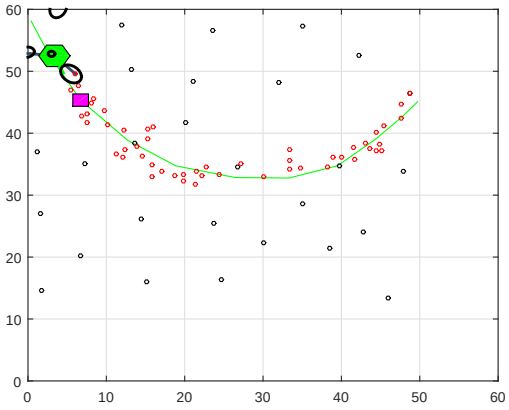


(b) Second experiment scenario 2D

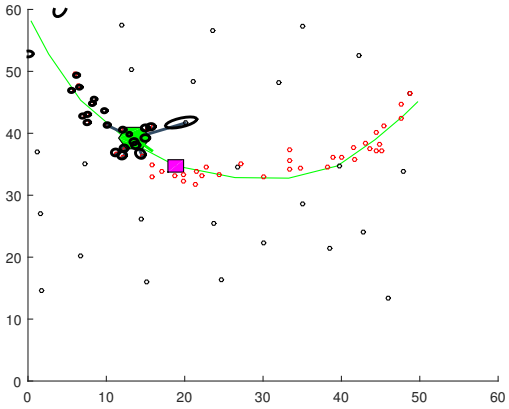


(c) Third experiment scenario 3D

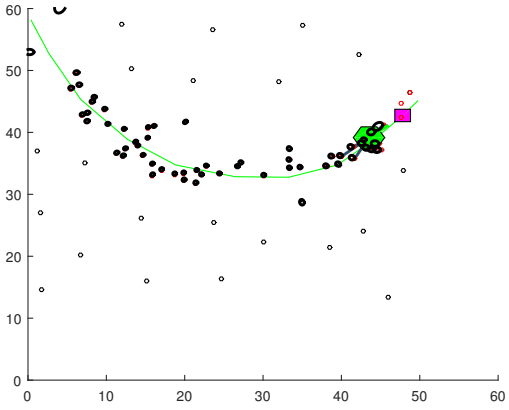
Figure 8.5: 6 DoF SLAM example scenario environments. The small black and red circles represent feature/features in the environment. The green lines represent platform trajectory. The shape right top corner represent the quad-rotor platform. The square pink box represent the next way-point.



(a) 2D view of map at $k = 660$

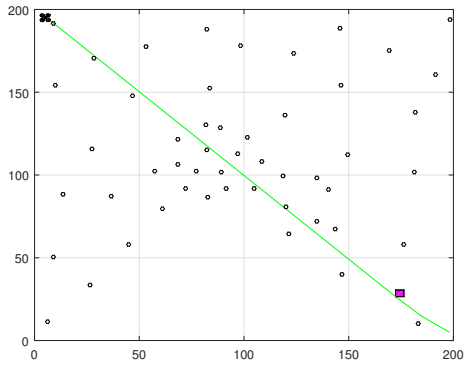


(b) 2D view of map at $k = 1220$

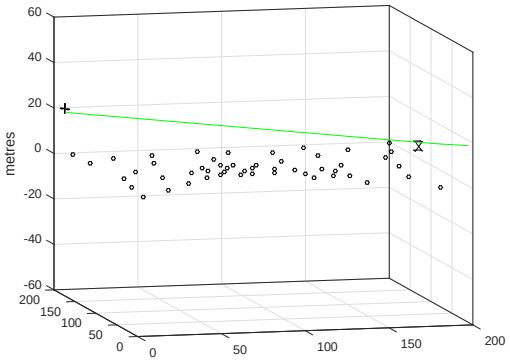


(c) 2D view of map at $k = 3030$

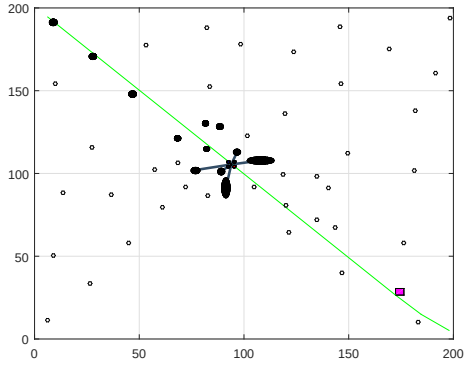
Figure 8.6: Plot 8.6a show the platform exploring the environment. The ellipses represent uncertainty in the position of initialised features. The green line represents platforms trajectory. The pink square represents the next way-point. The black dots represent features in the environment. The lines connecting the platform to features represent observations rays. Plots 8.6b and 8.6c show the SLAM constructed map of the explored environment.



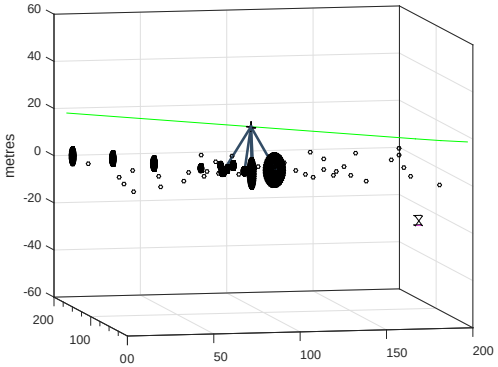
(a) 2D view of map at $k = 300$



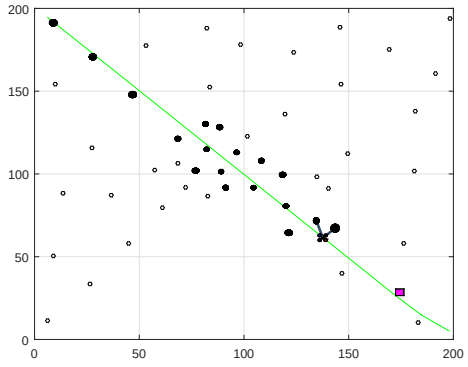
(b) 3D view of map at $k = 300$



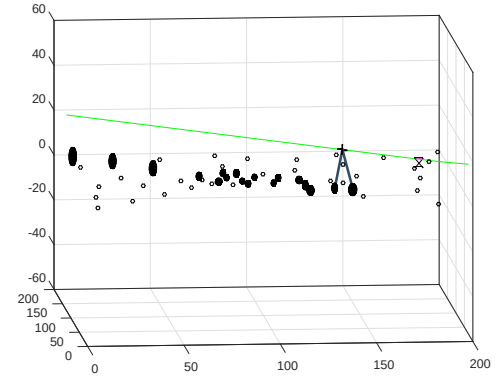
(c) 2D view of map at $k = 6750$



(d) 3D view of map at $k = 6750$

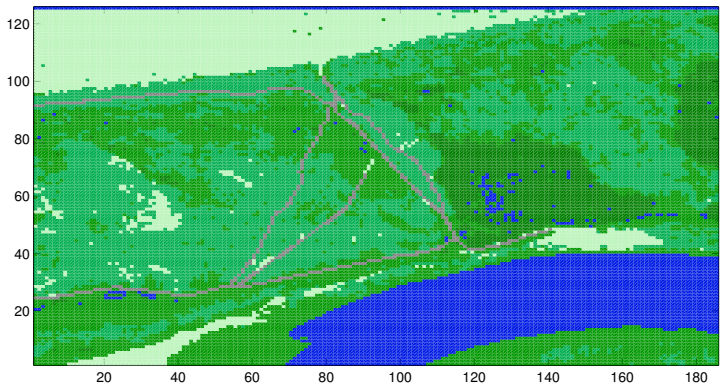


(e) 2D view of map at $k = 11450$

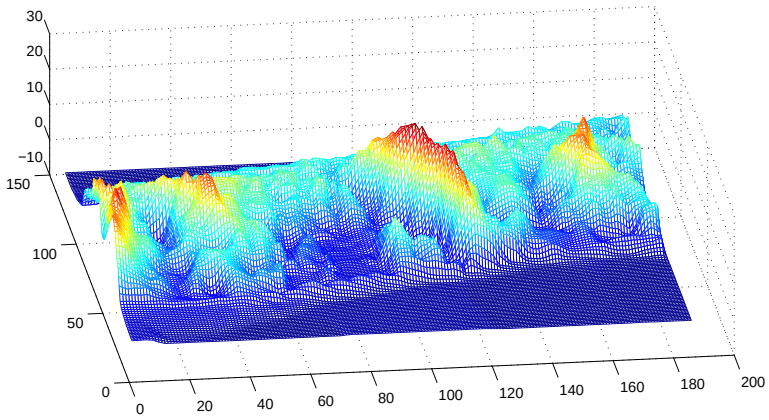


(f) 3D view of map at $k = 11450$

Figure 8.7: Plots on the right column show the SLAM generated map at different time-steps in 2D. Plots on the left column, show the SLAM generated map at different time-steps in 3D. Plots 8.7a and 8.7b show the platform at the beginning of exploration. The green line represents platforms trajectory. The pink square represents the next way-point. The spheres represent uncertainty in the position of initialised features. The lines connecting the platform to features represent observations rays.



(a) Classified model



(b) Elevation model

Figure 8.8: Sand Dunes data sets for the complete search model experiment.

8.5 Complete Search Model Evaluation

The aim of this experiment is to bring together all the elements of our proposed search model including SLAM, and show that it is able to perform well when compared against a grid-based search, even in the presence of uncertainty in the position of the UAV and in the position of the detected evidence features.

8.5.1 Scenario

We perform two experiments in this section. In the first experiment we will repeat the experiment performed in Section 7.4 (using the data sets for the New Forest) but with SLAM used to localise evidence features and the search platform.

To show that the search model can be used in different environments, we perform a second experiment. This experiment also consists of repeating the search steps for a scenario using the data sets for the Sand Dunes, as shown in Figure 8.8.

While the first experiment setup is exactly as given in Section 7.4, for the second experi-

ment we assume the LP has been missing for 20 minutes. Therefore, considering the assumptions about LP's walking speed $0.7m/s$, each agent is run for 2400 steps. This represents a maximum distance of 1680 metres. To generate the initial distribution using the diffusion based model, again considering our findings in Section 5.5.5, we ran the diffusion based initial distribution generation for 1344 steps, which represents around 80 minutes, four times the actual time.

Based on the results in Appendix F.5.5.1, we will use the DI implementation of 6 DoF SLAM with a baseline of 28° to initialise detected evidence. To enable the rapid localisation of features, we have oriented the vision sensor pointing downwards.

To ensure optimal performance, observed evidence is only used in the update phase of search model when the maximum eigenvalue of its covariance representing the uncertainty in its Cartesian position is less than 1 along all three axes (x, y and z). This means that the evidence will be well localised with an uncertainty of less than one metre in all directions.

8.5.2 Environment Setup

Due to time constraint, for both experiments assume we have recent data sets of the land cover. Therefore we do not need to re-classify the land cover, and hence no classification-based update is performed.

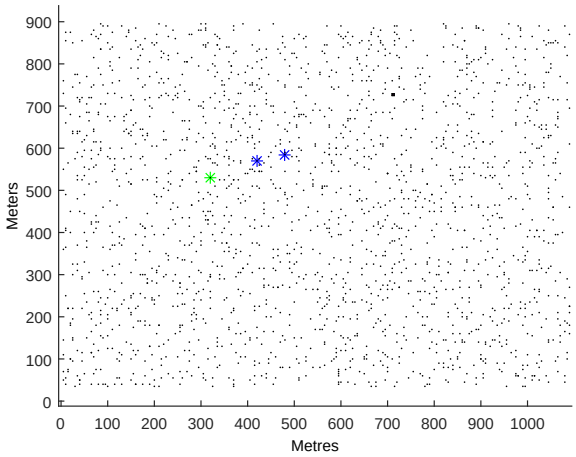
For the first experiment Figure 8.9 shows the point feature cloud representing salient features and evidence features. For this experiment we have simply sampled DSM of the search area randomly generating 2000 point features. The same applies to the second experiment. Figure 8.10 illustrates a typical UAV observation footprint.

8.5.3 Search Platform and Search Path

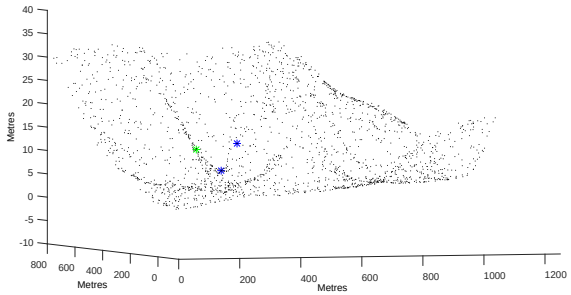
To simulate a quad-rotor UAV, we use a version of QRSim [29] with platform constraints as detailed in Section 1.2.4. Figure 8.10 shows how the simulator works.

The altitude of the UAV is kept at $20m$ above ground level. This is believed to be a good altitude for both control of the UAV, detection of evidence and detection of the LP. It provides a consistent camera footprint area of about $(15m)^2$. The constant altitude with respect to the surface of the terrain is archived by setting way-points, which consider the Digital Surface Model (DSM) of the search area.

In this experiment, and similar to the example in Section 8.4.1.6, we assume that the GPS operates at a rate of 10Hz as the prediction happens at a higher rate of 25Hz. The IMU sensor runs much faster at 30Hz.



(a) View from top



(b) View from and angle

Figure 8.9: The point cloud representation of features in the search area. The blue stars represent the evidence deposited by the LP and the magenta star represents the final location of the LP.

8.5.4 Evaluation Metrics

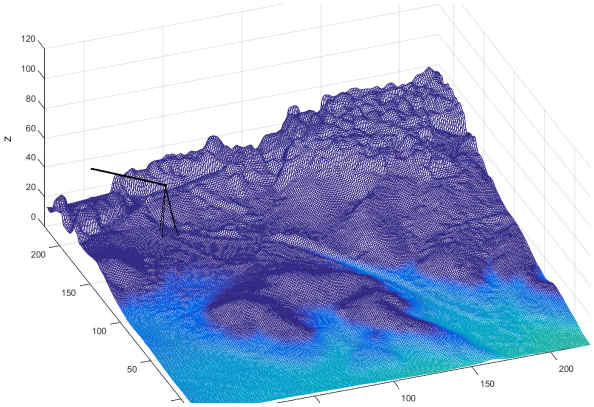
We will use the same comparison criteria used in the experiment in Section 7.4. The time-to-locate to evaluate the performance of the complete model with uncertain information and compare its results with the results achieved in Section 7.4.5.

8.5.5 Evaluation Results

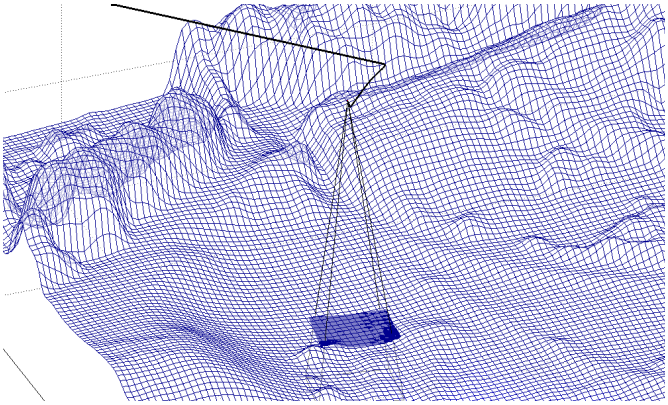
8.5.5.1 First Experiment Results

To show how the update process takes places using the complete search process, we once again consider a search for the LP that has traversed the environment following trail 2, illustrated in Figure 7.4. The evolving distribution over the trajectory of the LP is shown in Figures 8.11. The detected and localised features at the time the first evidence is detected is shown in Figure 8.12. As it can be seen the features that are observed repeatedly have a greater accuracy than the newly initialised features. Figure 8.13 shows the localised features at the end of search - when the LP is detected.

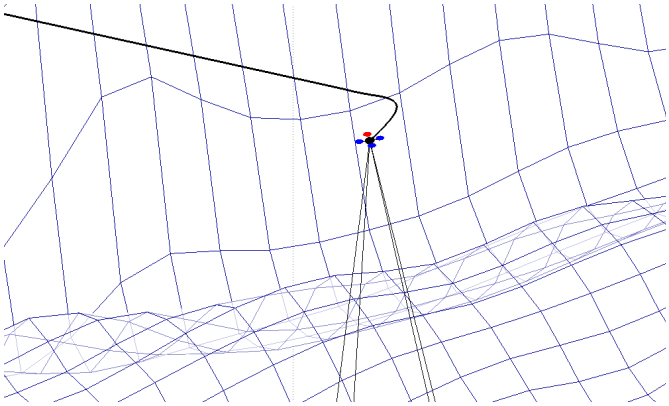
Comparing the entropy plots in Figure 8.14 with those in Figure 7.10, we can see that



(a) Quad rotor with Elevation data set

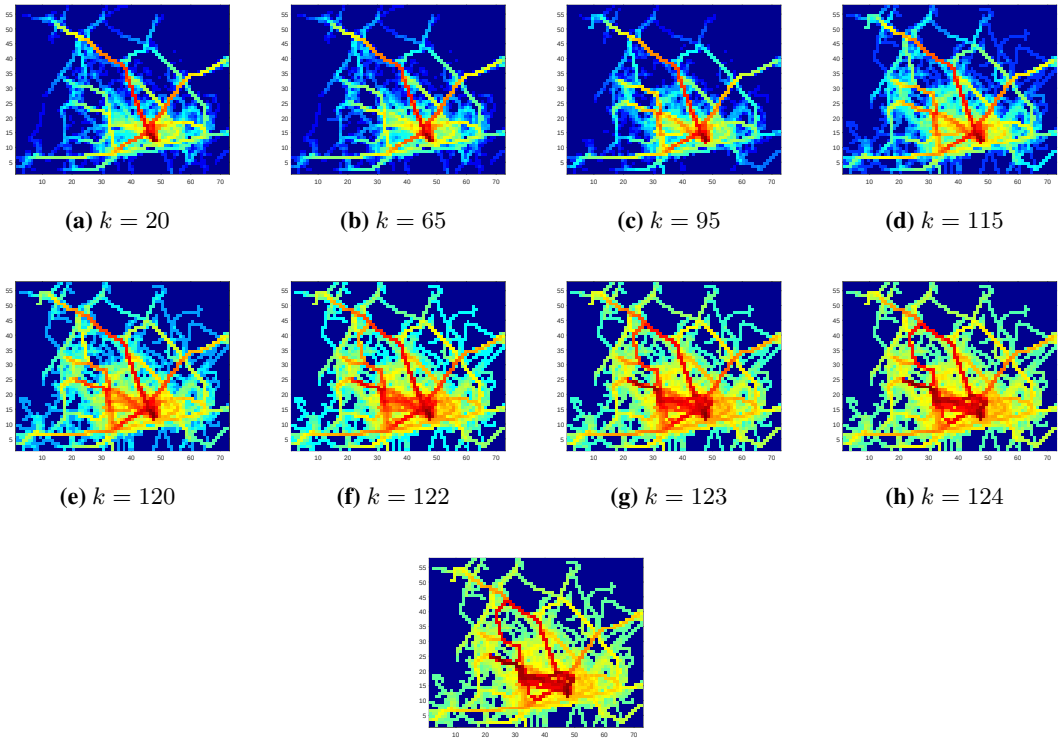


(b) Camera footprint



(c) Quad rotor close up

Figure 8.10: Plots showing the workings of Quad Rotor Simulator.



(a) $k = 20$ **(b)** $k = 65$ **(c)** $k = 95$ **(d)** $k = 115$

(e) $k = 120$ **(f)** $k = 122$ **(g)** $k = 123$ **(h)** $k = 124$

(i) $k = 125$

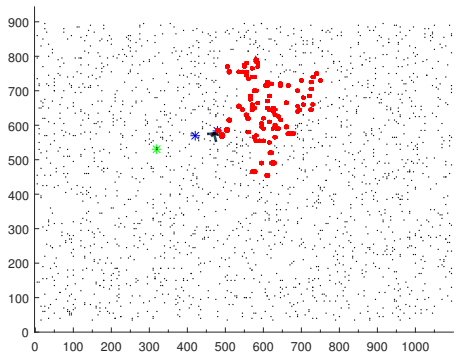
Figure 8.11: Log of evolving distribution over the LP trail using the complete search model.

on average, there is very little difference between the times it took to locate the LP with both known environment and unknown environments. The later of which has uncertainties in both platform and feature locations. This is because using the DI implementation of 6 DoF SLAM, with camera facing down, the model was able meet the baseline required for initialisation of detected evidence quite quickly. From this we can deduct that even with unknown environments and uncertain evidence location, the proposed search model is able to out-perform grid-based search model. This is specially true when the grid-based search is performed using a diffusion-based initial distribution.

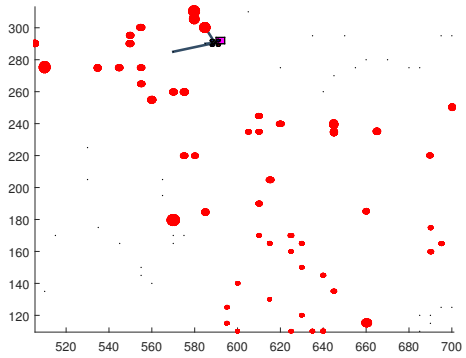
An advantage of using SLAM with Delayed Initialisation is that it allows for several observations of a detected evidence or the LP from different angles before its position can be initialised in the map of the environment. Which, in addition to reducing the uncertainty over the position of the feature, helps to reduce the probability of false classification of either the evidence and the LP.

8.5.5.2 Second Experiment Results

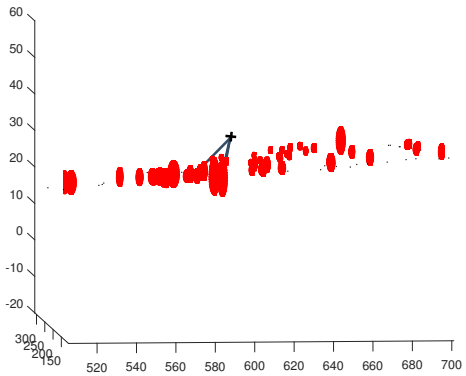
Figure 8.15 shows the scenario in the second experiment. Plot 8.15a shows the generated agent trail samples for a simulated time of 20 minutes. Plots 8.15b and 8.15c show the resulting



(a) Top view



(b) Top view closeup



(c) Side view closeup

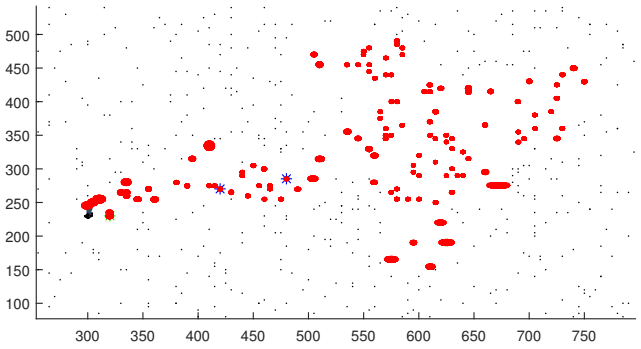
Figure 8.12: Different views of the localised environment features at detection of first evidence.

distribution over the LP trail and the end location respectively. Plot 8.15d shows the generated diffusion based initial distribution for the same scenario. In the second experiment, as can be seen in the entropy plot in Figure 8.16, once again our proposed search model completely outperforms grid-based search in all forms. This holds even when no evidence is considered in the update of the distribution or when only the agent-based distribution over the end location of the LP is considered. This is visually evident from Figure 8.17, which illustrates the UAV search path for different configurations.

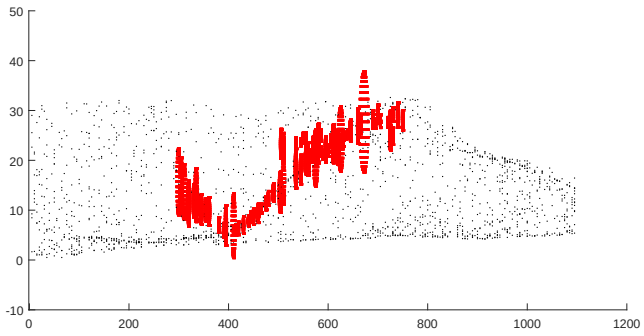
8.6 Summary

The focus of this chapter was to bring together all of the elements of our proposed search model and show how it performs against an existing grid-based search in the presence of uncertainty in both the search platform's position and the feature positions.

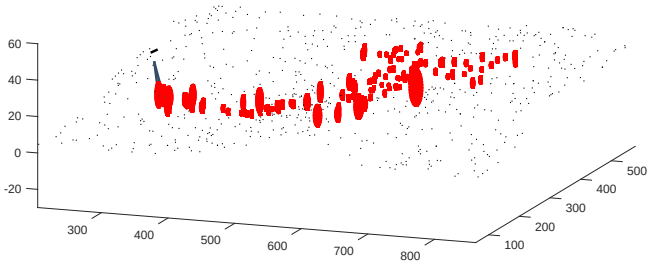
We started the chapter by investigating localisation methods. From existing methods, we decided to use a Kalman Filter (KF) implementation of a feature-based localisation method. However, since the search platform we are considering for this research is a quad-rotor, and be-



(a) Top view

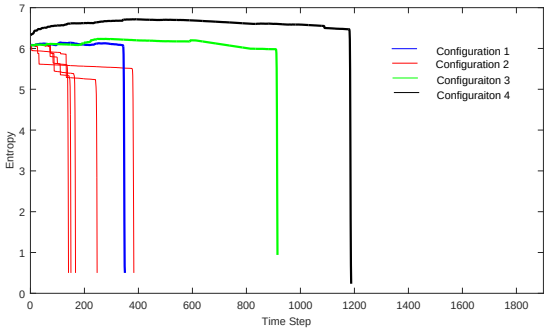


(b) Side view

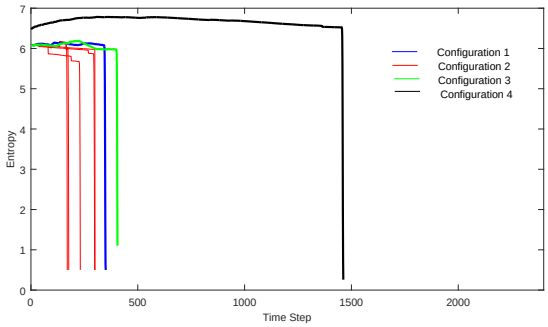


(c) Top view closeup

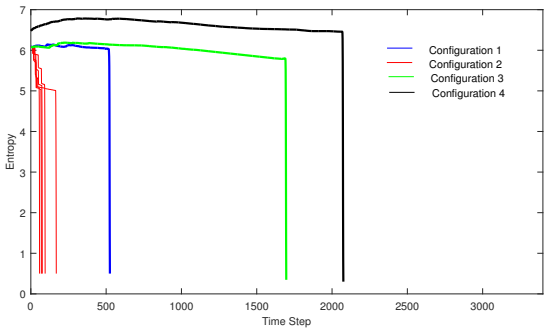
Figure 8.13: Different views of the localised environment features during search up until the LP is detected.



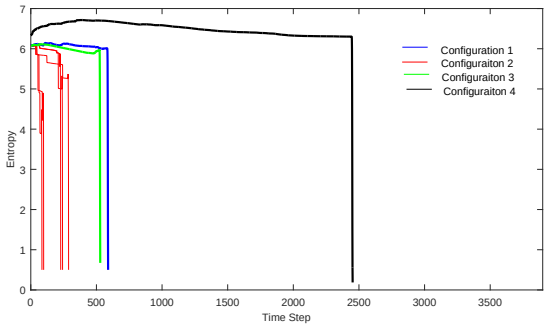
(a) Trail 1



(b) Trail 2

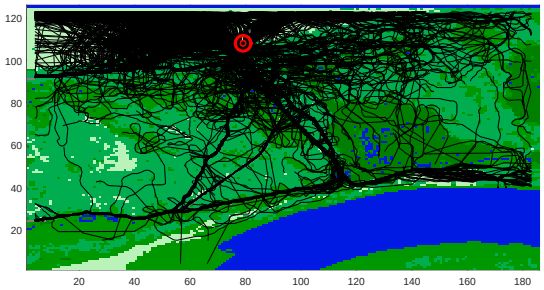


(c) Trail 3

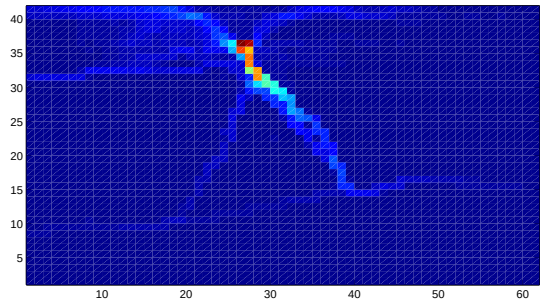


(d) Trail 4

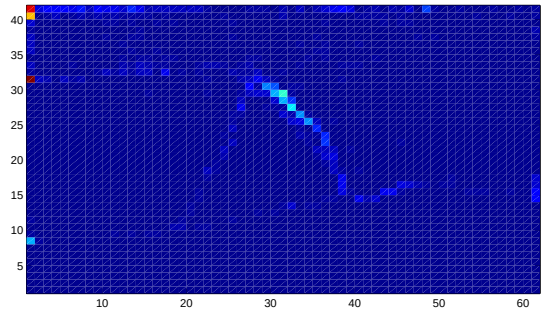
Figure 8.14: Comparison of the entropies computed during the life of the search using the proposed search model (with SLAM) and grid-based search models. Details of different configuration can be found in Section 7.4.5.



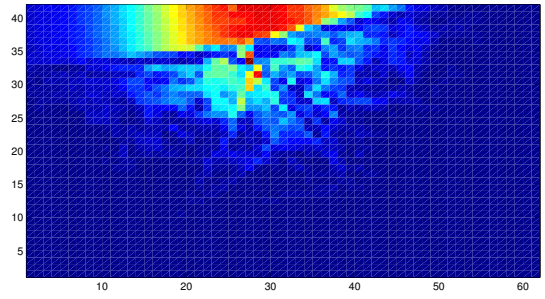
(a) Agent trail samples



(b) Initial distribution over trail



(c) Initial distribution over end location



(d) Diffusion-based initial distribution (80 mins)

Figure 8.15: The agent-based initial distribution generated over LP trail in a location in Northern Ireland. The red circled star in plot 8.15a represents the PLS. The discretisation of distributions reflect the camera footprint.

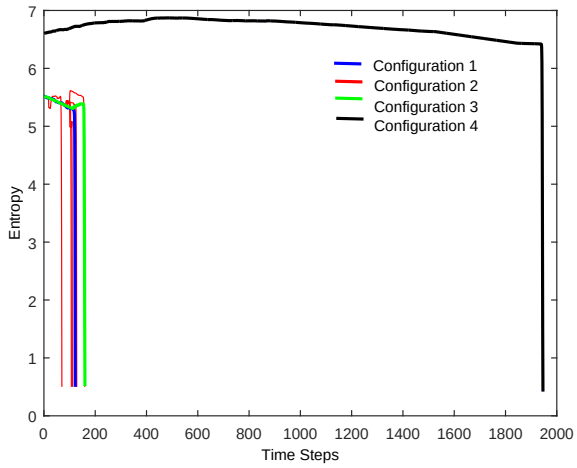
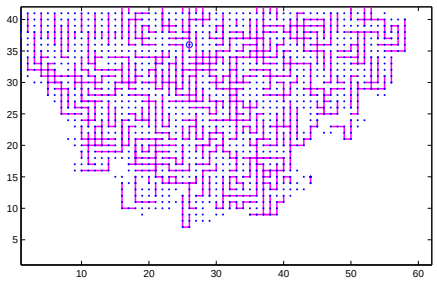
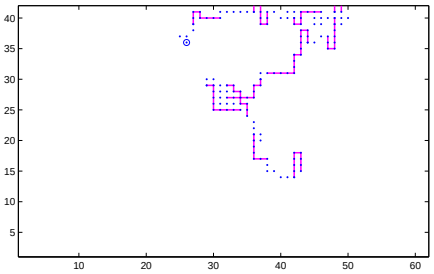


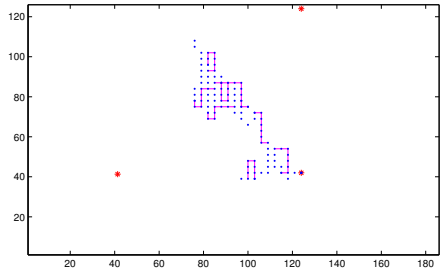
Figure 8.16: Comparison of the entropies computed during the life of the search for a LP using the proposed search model (with SLAM) and grid-based search models.



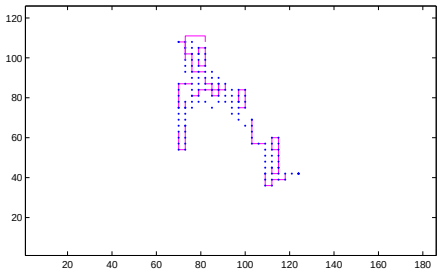
(a) Search path for configuration 4



(b) Search path for configuration 3



(c) Search path for configuration 2



(d) Search path for configuration 1

Figure 8.17: The UAV trajectory for the different search configurations.

cause it can only use one visual sensor to observe the environment due to the pay load limitation of the platform, we posed the localisation problem as 6 DoF Monocular SLAM with Delayed Initialisation of features.

Having decided on the SLAM implementation we run an experiment to evaluate the performance of the proposed search model as a whole in Section 8.5. From the results it was shown that even with uncertain platform position and feature location in the environment, the proposed search model was still able to out-performed the conventional grid-based search and produced similar results to Section 7.4.

The main drawback of the model is the additional computational complexity introduced by SLAM. However, this issue can be addressed by using more recent and advanced SLAM and memory management techniques.

Chapter 9

Summary and Future Work

This thesis investigated the problem of a search for a Lost Person in the wilderness using an autonomous Unmanned Aerial Vehicle (UAV).

Autonomous UAVs offer an unprecedented opportunity to revolutionise many applications, one of which is search and rescue missions. This is mainly due to the significant advantages they provide to search mission over ground-based search and piloted vehicle assisted search. These advantages include agility, expandability, reduced purchase and running costs.

Within this research we have sought to formulate and develop a search model that consists of conventional search phases: *information gathering*, *initial distribution generation* and *search*. The key differences to current automated search models are in both the way the initial distribution is generated and the update is performed during the search phase:

- *Initial distribution generation*- This is computed over the LP's trail rather than the end location using an agent-based LP movement model, which considers the LP dynamics, behaviour and long range interaction with the environment.

- *Update* - In addition to LP location information we consider evidence and new land cover information. This informs the search model and updates the estimate of the LP's whereabouts.

The flow of the proposed search process along with its coverage in this thesis is illustrated in Figure 9.1.

9.1 Summary

After presenting the context for this research in Chapter 1, in Chapter 2 we a presented overview of the WiSAR. We described the key phases of the search for a lost target in WiSAR. These are *information gathering*, *initial distribution generation* and *search*. We then investigated the current initial distribution generation methods and search models. We focused on a diffusion-

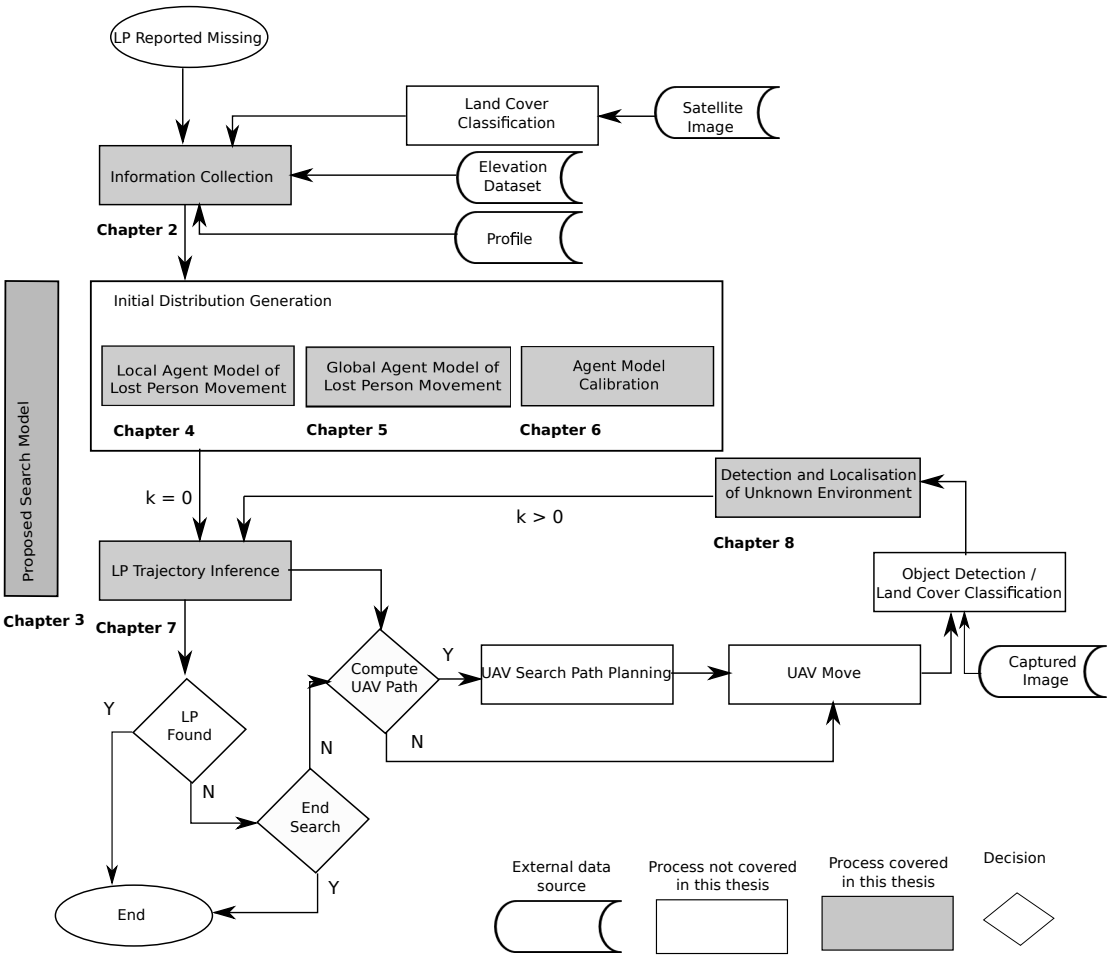


Figure 9.1: Flow diagram showing the proposed search process and its coverage in this thesis.

based initial distribution method and a sequential grid-based search model. Both are considered state-of-the-art in search literature. Upon investigation it was clear that both suffer from several critical drawbacks. The diffusion-based method marginalises the LP's movement history, modelling only the LP's local interactions and orientation strategies, and ignoring their state dependent behaviours.

Similarly, the grid-based sequential search model only considers the LP's location information, ignoring other information types typically used in WiSAR, such as evidence that may have been deposited by the LP or new land cover classification.

To address these limitations we proposed a new search model in Chapter 3. We detailed the proposed search model using a graphical model and discussed the information types required for the model to work along with the process of search using it. A crucial part of the proposed search model is the generation of an initial distribution that is a faithful representation of the LP movements. In order to generate this initial distribution, we needed a formal way to consider the LP's behaviour and the long range interaction with the environment.

Therefore, first, we proposed to model LP's movements and long range interaction with the environment using ABM. Then we formulated a probabilistic model of LP's trail. Since the LP dynamics can be very non-linear, we needed a more expressive way to model it. As a result, we decided to sample the distribution using particles. Each particle was modelled using an agent with sampled behaviour parameter values to capture the uncertainty in the LP's behaviour.

Since implementation of the agent model can be very complex, we covered it in two chapters. We started by presenting the general design of the agent model and implementation details of the agent model with only local interactions in Chapter 4. We then improved on this agent model and introduced global interaction capabilities and a number of state dependent behaviours and re-orientation strategies in Chapter 5. The proposed agent model of LP's movement offers several advantages over diffusion-based initial distribution:

1. With an ensemble of agent trails, the probability distribution over both the LP's trail and the end location can be computed. We showed that this is a very helpful when inferring the LP's direction of travel during the search phase.

2. Due to its state dependency, agents allow modelling a number of reorientation strategies performed by lost people such as direction travelling and behaviours such as fatigue, speed variations. Through experiments, we showed that each of these help generate a more faithful representation of the LP's trail.

3. Compared to the distributions generated using the diffusion model, which simply diffuses the distribution over LP location in all directions, the distribution generated using the agent trails can be sparse, only over potential trails of the LP.

To determine the significance of the agent-based human movement model, we performed an experiment using the data sets from an area in the New Forest In this experiment, we compared the distributions generated using both the agent and the diffusion models to distribution of the GPS logs of participants movements gathered in a data collection experiment held in the same area of the New Forest. We found that by considering the LP's state dependent behaviours, re-orientation strategies and long range interaction with environment, the agent model of the LP's movement was better able to represent the observed data. In addition, we identified that the diffusion model required to use four times the actual time the person has been lost in modelling the distance travelled by the LP. Also since the distribution is spread in all direction, and therefore, it is not able to produce good estimates at far locations.

In the second experiment, we put this outcome to the test. We compared and evaluated the use of agent model, diffusion model, uniform, Gaussian, and feature-based distributions as

initial distribution in a grid-based search for a target. With a simulated quad-rotor searching the environment cell-by-cell, using the look-ahead path planning, we showed that using the agent-based initial distribution over the LP trail, the quad-rotor was able locate the LP in significantly shorter times, visiting fewer cell compared to the case where the initial distribution used for the search was diffusion-based or any other type.

There is however one drawback in using ABMs. ABMs models consist of many parameters, the values of which are set either heuristically or based on search literature using distributions with high uncertainty to model the lack of knowledge. To tune values of these parameters, they need to be calibrated. This was done in Chapter 6. Since the number of parameters used to model the various behaviours were many, we decomposed the calibration challenge into two steps: pre-calibration step of parameter screening and the the calibration step. In parameter screening, we used a fractional factorial Design Of Experiment (DOE) to identify sensitive (or important parameters). Then in parameter calibration step, using a novel calibration method called SMC^2, we calibrated the identified sensitive parameters using the remaining observed data[1].

Having calibrated the agent model parameters, we needed to show that the agent model developed was able to faithfully represent the LP's movements and interactions with environment. To do this, we detailed a couple of validation frameworks. Each made up of a number of phases. We argued that, while we had performed most of the steps in the validation phases in each of the framework during the development of the agent model, we could only do indirect validation against real world observed date due to difficulty of getting real world data. This was done by performing the second experiment in Chapter 5 but with calibrated agents. Comparing the results achieved, with those for the second experiment in Chapter 5, we showed that using the calibrated agent generated distribution, search times were nearly halved. We interpreted this to mean that the calibrated agent model was able to better represent movement of data collection experiment participants compared to non-calibrated agent model. Thus the UAV was able to identify the path taken by the target much quicker.

Having completed the initial distribution generation task, to take advantage of the fact that LPs tend to drop evidence when they walk and that the environment model used to generate the initial distribution may not be current, in Chapter 7, we formulated a probabilistic update model that considered both positive and negative observations of the evidence deposited, the LP, and new land cover classification. We covered this in two steps. First, we modelled the LP

[1]Observed data was divided into two sets: one set used for evaluation of agent generated trails and the other for calibration.

and evidence observations. Then we modelled new land cover classification. To evaluate this new update model, we performed two different experiments respectively. In the first experiment we compared the performance of our search model with the grid-based search model in terms of entropy and time-to-locate metric. In the second experiment we evaluated affects of re-classification of the land cover on the inference of the LP trail and time-to-locate metric. Results for the first experiment showed that considering evidence features and the LP trail information as oppose to only the LP end location information can help reduce the search times by a factor of 2 to 25. Even if the LP is not detected, his travel path can be inferred, which the ground search team can investigate further. Similarly, results for the second experiment showed that considering new classification of the land cover can further reduce the search times by a factor of 1.5 to 3. Therefore, we concluded that using both evidence and land cover information in addition to the LP location information is vital to efficient and quick search.

This far, we assumed that throughout the search operation, both the platform and the detected evidence features are localised – there exact coordinates are known. However, in reality this is hardly the case. Even with sensors such as GPS on-board the platform, it cannot always localise itself due to factors like line of sight requirement. To deal with this issue, we investigated a localisation method called Simultaneous Localisation and Mapping (SLAM) in Chapter 8. Based on our system requirement – a UAV with payload limitation allowing to only add one monocular camera to it, we narrowed down the SLAM implementation to 6 Dof Monocular SLAM. This method has several initialisation techniques which play a significant part in the performance of the method. To identify the initialisation method that is able to generate consistent and stable map of the environment with good localisation of features, we performed a comparative study of three main techniques: DI, IDP and AHP. Result of the simulated experiment showed that DI performed consistent overall, providing with better localisation of features observed and of the platform's state.

Finally, having developed and tested all elements of our proposed search model, to assess the performance of the search model as a whole, we repeated the experiment in Chapter 5 using the complete search model. From results achieved, we showed that even with unknown environment and uncertainties in both the platform and feature locations, our proposed search model was able to outperform the grid-based search model.

9.2 Contributions

The main contributions of this research are:

- *Agent-based initial distribution generation over LP trail:* In this research in contrast

to existing models, we proposed and modelled the initial distribution over the LP trail, and not the LP end location. Since the LP dynamics and interactions can be very non-linear, we sampled it using particles, where each particle is an agent modelled to capture the LP behaviour and interactions with the environment. We showed that, using such a distribution can significantly improve search operation in terms of the search time and search efficiency.

- *Agent Model Calibration:* Our proposed agent model consists of many parameters. To tune there values, we used a two step process of sensitivity analysis and calibration. For the latter, we used a novel method called Sequential Monte Carlo Squared (SMC^2). In agent model literature, usually calibration is performed for models that consist of 5-8 parameters. However, using SMC^2 we showed that agent models with bigger number of parameters can also be calibrated. We showed that using the calibrated agent model we can generate more refined distributions that are more representative of the LP movements.

- *Update model considering extended information:* Once again in contrast to the search literature where only the LP end location information is considered during the search or the update phase, we proposed and used extended information: evidence features deposited by the LP and new land cover classification. We showed that considering such information types, can help refine the search area and infer LP trail significantly faster reducing search times by an order of up to 25.

- *Inclusion of a localisation method in modelling the search:* When performing automated search, mostly it is assumed that the platform and observed target are both localised at all times, which means there position is known accurately. However, this is rarely the case in reality. Therefore, it is imperative that the searching platform along with its observations are localised for two reasons. First, the platform requires its location information to decide where to go and search next or to report its location to ground controller. Second, the localised observations are required for trail inference. To deal with this, we incorporated a localisation method called SLAM into the proposed search model. We showed through experiments that this enables the search model localise the detected features and utilise them in the update phase even if there was some uncertainty in there position. Still out performing the grid-based search model.

9.3 Future Possible Improvements

There are a number of avenue of research that can follow on naturally from the work in this thesis.

- *Implementation language:* Current system was implemented using MATLAB in a functional form. As a result, the processing times for both agent model and SLAM grew increasingly expensive. While this implementation was sufficient for the proof of concept, which this thesis was aiming for, the next step and a very important step would be to re-implement the proposed system using an object oriented language such as Java or C++ utilising there multi-threading properties or libraries for parallel operations such as MASON [142]. These languages and libraries will provide the agent model the modularity useful for developing a computer simulation and help reduce the processing times, allowing for real-time test of the model using a UAV platform.

- *Nature of the search area:* Because wilderness consists of different types of topography, such as plain fields, tall trees, and mountains, there are many potential places an LP could rest, wait or hide (i.e. under canopy of trees). For this research however, we considered the case where the LP is visible from a UAV searching the area. The next step would be to extended the model for other ground types such as forested areas, where the UAV is not able to observe paths through the dense forest. For these scenarios, the movement of the LP can be modelled by a random walk in forested regions of the search area.

- *Number of parameters in the agent model:* The agent model designed and developed to model the LP movement contains many parameters. While these parameters were screen and identifying sensitive parameters in Chapter 6. Due to time constrains, we could not investigate reducing the number of non-sensitive parameters. Therefore, a future improvement to the model can be to investigate reducing these parameters and study effects of it on the agent movement. This can be achieved by checking the agent model with a verity of scenarios and terrain types such as temperate, dry domains with different configuration of environment feature classes and identify non-sensitive parameters that have negligible affect on agent movements.

- *Object detection and recognition:* In this research we represent evidence features as point features and assume that they can be detected and classified. Therefore another potential area that requires investigation is classification of land cover and detection of people, and evidence features from actual UAV captured information. This can be done by using computer vision methods and modelling classifiers.

- *Consider new classification of land cover with complete search model:* Due to time constrains, in the final experiment, we assumed the classified model of the land cover was current. Therefore we used the feature-based SLAM not considering new observations of land cover classifications. A future improvement to the model could be to incorporate a variation of the SLAM such as Hybrid Metric Map (HyMM) [217], grid-based SLAM [194] or method proposed in [218] that allows for localisation of polygonal shaped regions, which can be used to model changes in the environment. In the later approach, triangular and point-like objects are represented within a single data structure. This provides a natural way to support both point-like and area-like features (land cover classification represented by polygons).

- *Considering varying weather condition:* Considering the searching platform constrains (low battery power) and typical size of a search operation (few hours), in this research, we assume that weather stays constant in wilderness for the duration of search. However, in reality, the weather can change and the search operation can take longer than few hours. Therefore, to relax this assumption of static weather, an improvement to the model can be to extend it to deal with varying weather conditions. To achieve this a model for the time evolution of the search area weather needs to be specified and applied at each step during the computation of the LP trail.

9.4 Future Possible Experiments

Due to exciting nature of this research and its potential to achieve so much more, there are a number experiments that can be performed. This includes:

- *Open world assumption:* At the time of search initiation, a LP can be in one of two states:

  - *Static:* The LP has decided to stay put, has reached an intended location not known to others or is injured and cannot move.
  - *Mobile:* The LP is mobile and continues traversing the area.

  In this research we considered the first category of lost people. Therefore, an obvious next step would be to test the search model with moving target and thus an expanding search area. This can be achieved by continually assessing the distribution over LP for the presence of the LP in the defined search area. If the LP is deemed to be outside the defined search area, a possible step in search can be to extend the search area and as a result extend the distribution over LP to cover the extended region of the search area. This

can be achieved by propagating the agent particles forward in time to cover the additional travel time of the LP.

- *Collecting GPS log of lost people:* The GPS logs used to calibrate the agent model were collected during a data collection experiment, where participants of the experiment had to traverse an area in New Forest with wilderness like features such as variation in vegetation density, elevation, paths and obstacles. There task was to find a place (a car park) location of which was unknown to them. They had to use environment cues to orient and locate the car park. Although, the environment offered much of the diversity found in wilderness such as elevation change, vegetation density changes, occluded areas, and paths, the data collected might not be entirely accurate representation of a LP movements in wilderness due to following reasons:

  - *Small size of the area:* The area selected for the experiment was about $1km^2$. This size of area might not be sufficient to excite the complete set of behaviour exhibited by lost people.
  - *Participants were not lost:* As a result of this, effect such as anxiety, distress, and dehydration could not be captured.
  - *Participants were all aged between 20 and 35*[2]*:* As a result of this, the data did not represent younger or older group of people.

  Although we have shown that the agent model parameters can be calibrated using the novel Sequential Monte Carlo Squared (SMC^2) method, which was one of the objectives in this research. We understand that for the agent model to be well calibrated, the calibration should happen using good number of real lost hiker movement data, which will capture all behaviours exhibited by lost hikers in wilderness.

- *SWARM search:* In this research we considered using a single simulated UAV to perform the search operation. Although, this was sufficient for proof of concept. To show the potential of the proposed search model, specifically benefit of using agent generated initial distribution on LP trail, it would be very interesting to perform an experiment with several UAVs. Each searching different region of the search area prioritised using the underlying agent-based distribution. The coordination of the platforms could be modelled using decentralised methods such as the one proposed in [31].

[2]Participants younger or older were not selected because of ethical and safety reasons respectively

In Chapter 5 and then in Chapter 8 we showed that by using an LP trail-based initial distribution, we can search and locate the target much quicker than conventional distributions. We believe, this efficiency can be further improved by exploring different hypotheses that can be made using the agent-based initial distribution.

- *Search for different category of Lost Person:* Although there are around 33 categories of lost people, in this research we have considered only the lost hiker category (the dominant category of lost people). Having said this, with little modification to the parametrisation, we believe that the agent model can be used for other categories of LP such as runners, hiking campers and people with dementia (Alzheimer). Thus, another good experiment would be to consider a different LP category, re-parametrise the agent model for that LP category, run the search model and evaluate the performance of the system. Also, since in this thesis, values for agent behaviour parameters are set either heuristically or based on search literature such as [3]. An obvious extension to the experiment can be asking an experienced searcher to set these values.

Appendix A

Data Set Pre-Processing

A.1 Discretisation Resolution

Working with grid representation of search area, a key issue is deciding the resolution of the grid cell. There is a trade-off between cell resolution and storage. As spatial resolution increases, cells become smaller and number of cells scales quadratically in the gridded representation. The resolution of the discretisation used in literature varies depending on application. For example [219] uses $(4\text{m})^2$ for obstacle avoidance and path planning for a humanoid robots, [61] uses $(24\text{m})^2$ cells to ensure computational efficiency and [122, 123, 220] use much finer cells sizes of $(0.7\text{m})^2$ to model human step size.

For this research, we consider two important factors in determining the size of the cells. First, the cell size should not be so large that small features like paths would disappear. Second and vise-versa, the cell size should not be so small that it gets computationally too intensive to compute the model. For experiments we have performed in this thesis, we have found that cells of size $(5\text{m})^2$ meet both these conditions. This discretisation is used throughout this thesis unless otherwise specified

A.2 Data Set Pre-Processing

The resolution of the data sets representing the search area vary. For consistency and ease of use, each data set is pre-processed so that same cell size is used to both discretise the search area and the data sets representing it. In this research when acquired, while elevation data set E was of the right discretisation resolution ($(5\text{m})^2$), both vegetation V and topography T data sets were of higher resolution ($(25cm)^2$). To ensure the consistency of discretisation across all data sets, V was directly processed by joining each block of 20×20 in the original data set and setting its classification as the mode of the vegetation classes within the block. A synthetic example of this is illustrated in Figure A.1 with block size of 4×4. Similarly, T was also processed by joining each block of 20×20 original data set and setting its classification as the

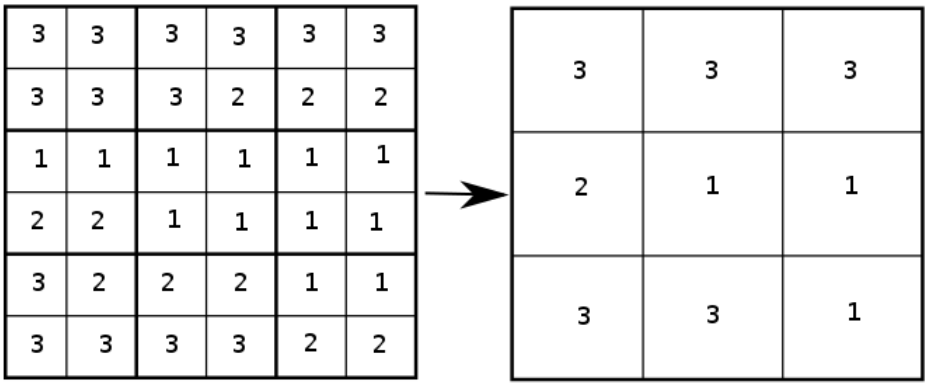


Figure A.1: Example of synthetic vegetation data set pre-processing. In this example, the data set is pre-processed in blocks of 4×4.

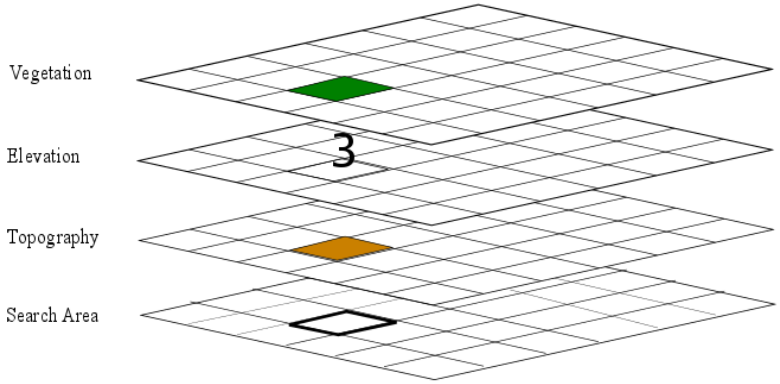


Figure A.2: Alignment of data sets representing the search area. A cell corresponding the a region within the search area has same coordinate in all data sets. This is achieved by aligning the raster representation of search area data sets using their UTM coordinates.

mode of the topology classes within the block with one exception: To accommodate topography features that are most attractive to LPs such as paths, if number of cells classified as path within the block in the original data set are more than a threshold (in our case 50), the block is set as path class irrespective of the mode topography class within the block. In case two or more topography feature classes happen to have same number of cells inside the block, the block is set as one of the classes at random.

Another important factor to consider when working with multiple data sets representing same area is the correlation of the data sets Defining a_i to be a cell corresponding to region i within the search area, the correlation and alignment must be such that a_i in each of the three data sets (V, T and E) must correspond to cell a_i in the search area illustrated in Figure A.2. Therefore, having correlated the data sets, each discretised region or cell within the search area $\mathcal{A}$ can be modelled as a tuple of V, T and E.

Appendix B

Search Paradigms

B.1 Manual Ground Search Paradigms

This includes:

- *Hasty Search*: Usually the first search paradigm used, it is used to search the high probability areas first.
- *Constrained Search*: Using this search paradigm, search teams attempts to locate any sign that may limit the search area. This is done by determining whether a LP has or has not crossed key areas such as a bridge. If it is determined that the LP has not crossed the bridge, the area on the other side of the bridge can potentially be eliminated as candidate for search.
- *High Priority Search*: The hasty and constraining searches often inform a high-priority region search, or priority search.
- *Exhaustive Search*: A less effective search paradigm compare to priority search. In this search paradigm, a systematic coverage of the area occurs using appropriate search patterns. For example, the search area is combed by search teams by forming a line and walking through the search area.

B.2 Unmanned Aerial Vehicle Assisted Search Paradigms

This includes:

1. *Sequential Operation/ Information Only*: In this search paradigm, a UAV is deployed to gather information independent of ground team. If a valid sign is found, ground support is dispatched to check it out. This search paradigm can be used for cases where terrain offers limited ground mobility or when the probability of locating the LP is uniformly

distributed across the search area. It is also suggested for exhaustive search or search in low probability regions.

2. *UAV-led Operation*: In this search paradigm, the incident commander stays with UAV operator and sensor operator at a base usually located at PLS. The UAV is directly supported by ground search team. A flight path is selected by for example specifying way-points for the UAV to travel by. Both the UAV and ground search team travel along this path. The UAV performs spiral or sweep search of the area around the defined path and progresses together with the ground search team. As soon as a potential sign or evidence is spotted by the sensor team, the ground search team is instructed by the incident commander to find the location of the evidence and investigate it. This information is then used to update the search plan and UAV flight path. This paradigm is suggested for cases when the terrain allows the ground search teams to be highly mobile but when enough information is not available to perform hasty search.

3. *Remote-led Operation*: In this paradigm, a UAV is deployed to increase ground teams visibility by following ground team during hasty search tracking foot prints, scent trails, etc. This paradigm is suggested in cases, where the incident commander has good awareness of the area. In this paradigm, the UAV flies in orbits centred at the location of ground search team increasing their visibility without corrupting the trail.

Appendix C

Particle Representation

In general there are two techniques to represent a distribution: *parametric* and *non-parametric* representations. Parametric methods seek computational tractability by parameterising the probability distribution functions. For example, KF introduced by R.E.Kalman in [221] parameterises probability densities by their first and second moments. The underlying model attempts to first estimate the state $\mathbf{t}_k$ of a discrete-time controlled process that is governed by linear stochastic difference equation using

$$\mathbf{t}_k = \mathbf{A}_k \mathbf{t}_{k-1} + \mathbf{B}_k \mathbf{u}_k + w_k, \tag{C.1}$$

where the matrix $\mathbf{A}_k$ relates the state at time step $k-1$ to the state at step k, $\mathbf{B}_k$ relates the control input $\mathbf{u}_k$ to the state $\mathbf{t}_k$ and w_k models the process noise. Then in the second step, update the state estimated using a measurement $\mathbf{z}_k$ with a linear relationship with the state variable by

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{t}_k + v_k, \tag{C.2}$$

where the matrix $\mathbf{H}_k$ relates the state vector $\mathbf{t}_k$ to the measurement $\mathbf{z}_k$ and v_k represents the measurement noise.

As a result, KF has been shown to be particularly effective for linear target tracking problems [222], efficiently approximating the Gaussian beliefs associated with observed targets providing optimal state estimate for linear system subject to Gaussian noise. However, in addition to complex nature of many real world problems that cannot be modelled using linear models, a number of problems such as non-linearities in equations that describe the physical systems and inaccurate or incomplete models of the underlying physical problem, require the use of non-linear models [223].

Models such as Extended Kalman Filter (EKF) [224] and unscented KF [225, 226] handle the non-linear estimation problems. However these model are still limited by representing

distributions by two moments (mean and variance).

For problems such as ours, where it is required to keep the history of LP movement, we need a model that is able to describe uncertainty in a more expressive way.

Non-parametric representations such as *grid-based* and *particle-based* are examples of such models. The grid-based method discretise the target space into a finite number of non-intersecting sub-spaces, with each subspace supporting a portion of an approximation of the distribution [61]. As we showed in previous chapter, this suffers from several limitations and is not capable of faithfully modelling the LP movements. In contrast particle-based methods such as SMC used within Bayesian framework often referred to as particle filters [100] or bootstrap filters [101] distribute discrete particles representing distinct hypotheses about the target state, across the target space. It is able to represent any arbitrary distribution. This is done by characterising higher probability density regions by relatively dense distribution of particles, and lower probability density areas by locally sparse particles distributions. Contrary to KF, which assumes Gaussian posterior distributions, particle filtering creates a sample-based representation of the entire probability density function [101].

Appendix D

Data Collection Experiment

D.1 The Advert

Dear All,

Get Lost. And be paid to do it.

Well close enough, we are going to setup an outdoor experiment in the New Forest. The aim of the experiment is to log GPS data of people who are not familiar with the area while they are traversing the natural landscape, hence participants would be asked to simply walk and find a target location in a bounded 1km by 1km area. Participants will be accompanied by a member of the team at all times, who has good knowledge of the area and who will carry the logging equipment and ensure no one actually gets lost. This logged data would then be used to study people's movement patterns and behaviour in natural landscapes and used to calibrate agents models used to generate distribution over a lost targets whereabouts. No personal information would be taken from the participants and their individual participation would not be shared with any third party under data protection. All data collected would be used anonymously in our reports.

The date and time for the experiment has not yet been finalised, but will depend on participant availability and local weather conditions. We are looking for around 10 participants, each of whom will be paid 10 per hour.

If you might be interested in taking part in this experiment, please let us know. We will contact all interested parties at a later date to confirm their availability, and arrange a suitable time for the experiment to take place. If you have any questions, please do not hesitate to contact me.

Best Regards,

Wallizada Mohibuillah

University College London w.mohibullah@cs.ucl.ac.uk

D.2 Safeguarding

This experiment would include taking participants (recruited from Southampton) to the selected location and then from a start point where all the participants would be based, each participant would be asked to find a target location (car park). This would happen one by one. Each of the participants would be accompanied by the researcher when they are traversing the natural landscape who would carry the logging devices that would log the GPS log of the participants movements, hence the researcher would simply follow the participant. However in cases where the participant is not able to find the target location, the experiment is stopped after 15 minutes of walking and both the participant and the researcher walks back to the base. No personal information or questionnaire would be given to participants to fill. All participant both at the base and walking would be accompanied by a member of the research team who will walk behind the participant.

D.3 Ethical Issues Related Preparations

The participant may get lost while traversing the area, to address this, each participant would be accompanied by the researcher, who has very good knowledge of the area and has surveyed the experiment area. Both participant and the researcher would be required to have charged and working mobile phones, in addition the researcher would have a very accurate map of the area, which is updated with the track log of there movements, hence they can simply track back to the base station.

The weather may get bad, it may start raining or snowing. In this case the experiment would be stopped and continued at a later convenient date. The participant may not be able to find the target location, in this case, after the 20 minute time limit has passed, the researcher and the participant would go back to the base station. The days may be cold and it may be very hard for the researcher to collect log of all the participants in one day, for this reason the experiment would be performed over several days, considering both the weather and participants availability. The participant may get hurt while walking, to address this issue, a First Aid box would be carried by the researcher at all times.

D.4 Informed Consent Form for GPS Log Collection in Research Studies

Please complete this form after you have read the Information Sheet and/or listened to an explanation about the research.

Title of Project: Understanding human reorientation patterns and movements. This study

has been approved by the UCL Research Ethics Committee (Project ID Number): 4339/001

Thank you for your interest in taking part in this research. Before you agree to take part, the person organising the research must explain the project to you. If you have any questions arising from the Information Sheet or explanation already given to you, please ask the researcher before you to decide whether to join in. You will be given a copy of this Consent Form to keep and refer to at any time. Participant's Statement

I ————————-

- Have read the notes written above and the Information Sheet, and understand what the study involves.
- Understand that if I decide at any time that I no longer wish to take part in this project, I can notify the researchers involved and withdraw immediately.
- Understand that such information will be treated as strictly confidential and handled in accordance with the provisions of the Data Protection Act 1998.
- Agree that the research project named above has been explained to me to my satisfaction and I agree to take part in this study.
- I understand that my participation will be logged while I am walking the site.
- I understand that the result of the GPS log will be published in reports and I will be sent a copy if I requested. Confidentiality and anonymity will be maintained and it will not be possible to identify me from any publications.
- I understand that I am being paid for my assistance in this research and that some of my personal details will be passed to UCL Finance for administration purposes.

Signed: ————————-Date:————————————-

D.4.1 Safeguarding During Data Collection

In order to make sure that all participants were safeguarded, we had arranged for first aid kit on the field. The environment was selected so that it was safe and did not contain cliffs or deep bodies of water. The principle researcher was a trained first aider. Each of the participants were given a map of the area in case of emergency. They also had their personal mobile phone (signal was confirmed) that could be used in emergency to call. Throughout the experiment, the participant was accompanied by the researcher who walked behind to not influence the participant's behaviour. More details of the advert for participants, consent and safeguarding are provided in Appendix D.

Appendix E

Design Of Experiment and Calibration

E.1 The Design Of Experiment Matrix

$$
\begin{bmatrix} d^{(1)} \\ d^{(2)} \\ d^{(3)} \\ d^{(4)} \\ d^{(5)} \\ d^{(6)} \\ d^{(7)} \\ d^{(8)} \\ d^{(9)} \\ d^{(10)} \\ d^{(11)} \\ d^{(12)} \\ d^{(13)} \\ d^{(14)} \\ d^{(15)} \\ d^{(16)} \\ d^{(17)} \\ d^{(18)} \\ d^{(19)} \\ d^{(20)} \\ d^{(21)} \\ d^{(22)} \\ d^{(23)} \\ d^{(24)} \\ d^{(25)} \\ d^{(26)} \\ d^{(27)} \\ d^{(28)} \\ d^{(29)} \\ d^{(30)} \\ d^{(31)} \\ d^{(32)} \end{bmatrix} = \begin{bmatrix}
-1 & -1 & -1 & -1 & -1 & -1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & +1 \\
-1 & -1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & +1 \\
-1 & -1 & -1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & +1 & -1 & +1 & -1 & -1 & -1 & +1 & +1 \\
-1 & -1 & -1 & +1 & +1 & -1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & -1 & -1 & -1 & +1 & +1 & -1 & +1 & -1 & -1 & -1 & +1 & +1 \\
-1 & -1 & +1 & -1 & -1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & +1 & -1 & -1 \\
-1 & -1 & +1 & -1 & +1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & +1 & -1 & -1 \\
-1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & -1 & -1 \\
-1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & -1 & -1 \\
-1 & +1 & -1 & -1 & -1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & +1 & -1 & +1 \\
-1 & +1 & -1 & -1 & +1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & +1 & -1 & +1 \\
-1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & +1 & -1 & 1 & -1 & -1 & +1 & -1 & +1 & -1 & +1 \\
-1 & +1 & -1 & +1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & -1 & +1 \\
-1 & +1 & +1 & -1 & -1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 \\
-1 & +1 & +1 & -1 & +1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 \\
-1 & +1 & +1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 \\
-1 & +1 & +1 & +1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 \\
+1 & -1 & -1 & -1 & -1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & +1 & +1 & -1 \\
+1 & -1 & -1 & -1 & +1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & +1 & +1 & -1 \\
+1 & -1 & -1 & +1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & -1 \\
+1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & -1 \\
+1 & -1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & -1 & +1 \\
+1 & -1 & +1 & -1 & +1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & -1 & +1 \\
+1 & -1 & +1 & +1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & -1 & -1 & +1 \\
+1 & -1 & +1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 & -1 & -1 & +1 \\
+1 & +1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & -1 & -1 \\
+1 & +1 & -1 & -1 & +1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & -1 & -1 \\
+1 & +1 & -1 & +1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & -1 & -1 & -1 \\
+1 & +1 & -1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & -1 & -1 & -1 \\
+1 & +1 & +1 & -1 & -1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 \\
+1 & +1 & +1 & -1 & +1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 \\
+1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\
+1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1
\end{bmatrix} \tag{E.1}
$$

E.2 Regression Result

Table E.1 shows the regression analysis result.

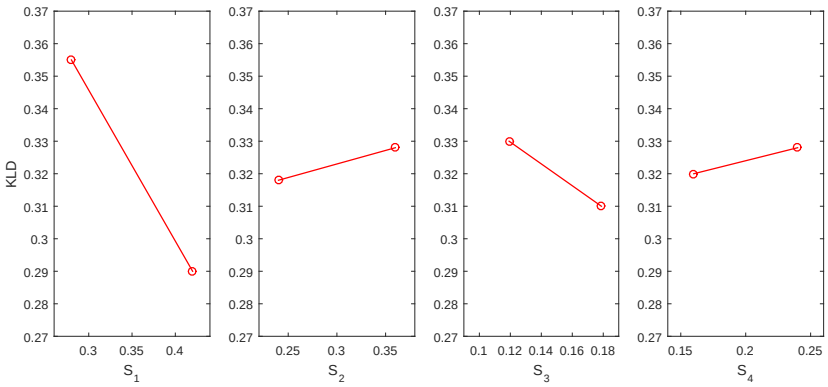
From this, we need to check that the individual coefficients are statistically significant. This means checking the null hypothesis that the model parameter coefficients could be zero, which would in return mean that the related factor may not have any affect on the response. We do this hypothesis test using the $p - value$. As can be seen 14 parameters have p_value less than 0.05, the significance level used. This means, we have strong evidence to reject the null hypothesis for these 14 parameters $(\{S_1, V_{(1,1)}, V_{(1,2)}, V_{(1,3)}, T_{(1,1)}, T_{(1,2)}, T_{(2,3)}, T_{(3,1)}, T_{(3,3)}, T_{(3,4)}, T_{(4,2)}, T_{(4,3)}, T_{(4,4)}, s_{max}\})$ and accept the null hypothesis for the rest.

E.2.1 The Main Effects Plots

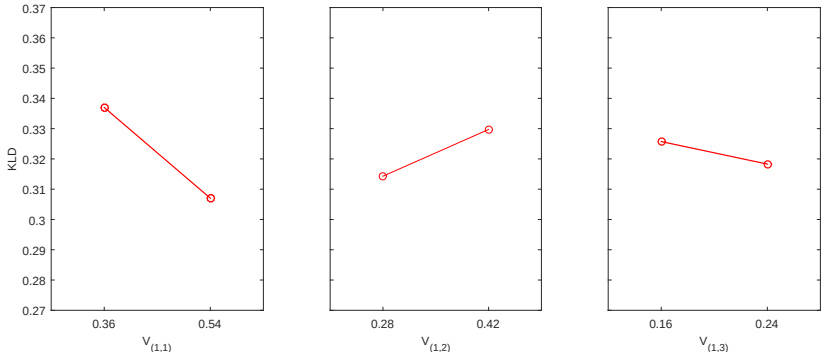
The regression result is reflected in the main effects plots in Figures E.1 to E.3.

E.3 Calibration Results

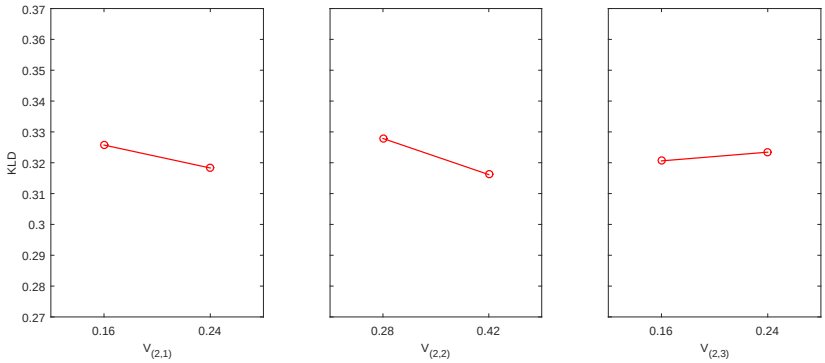
Figure E.4 shows the distribution over parameters both before and after the calibration process. As we can see, before the calibration process, because the distribution over each parameter was only specified by the two moments of the distribution (mean and standard deviation) using knowledge of search literature, the distributions are very smooth. However as a result of the calibration and due to the multi-modality of the distribution over $\mathbf{t}$ and changing behaviour of the participants in the data collection phase, the calibrated distributions are multi-modal with reduced uncertainty.



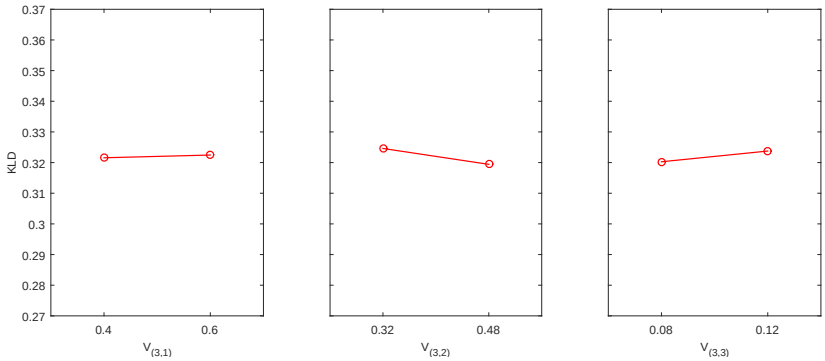
(a) Elevation sensitivity



(b) Vegetation sensitivity row 1

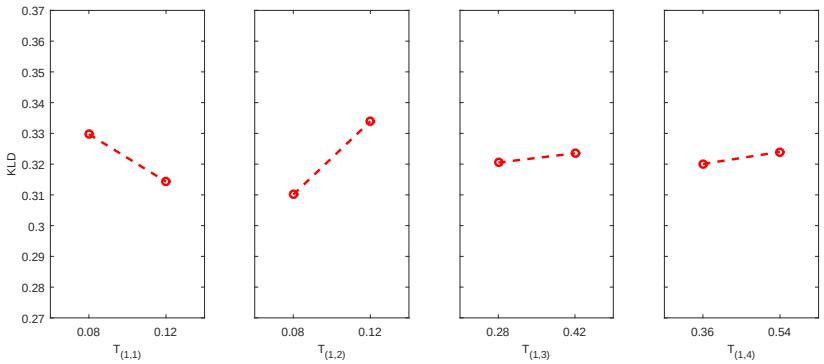


(c) Vegetation sensitivity row 2

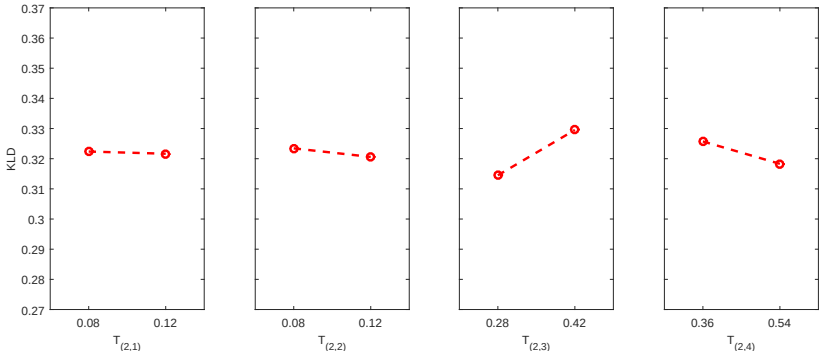


(d) Vegetation sensitivity row 3

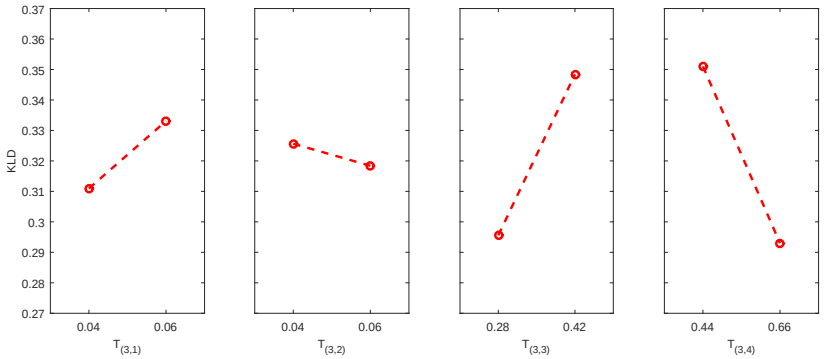
Figure E.1: Sensitivity analysis results (set 1).



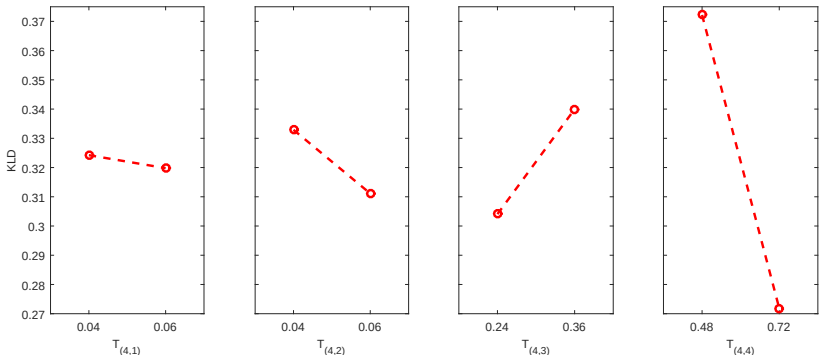
(a) Topography sensitivity row 1



(b) Topography sensitivity row 2



(c) Topography sensitivity row 3



(d) Topography sensitivity row 4

Figure E.2: Sensitivity analysis results (set 2).

Table E.1: Regression analysis results.

ANOVA

	df	SS	MS	F	$SignificantF$
Regression	30	0.791		27.33	1.41E-26
Residual	65	0.0627	0.001		
Total	95	0.8537			

Term	Coefficient	Standard Error	t Stat	P-Value
Intercept	0.52410	0.0869	6.03400	0.0000
S_1	-0.5098	0.0453	-11.259	0.0000
S_2	0.08360	0.0528	1.58180	0.1186
S_3	-0.2906	0.1064	-2.7320	0.0081
S_4	0.09500	0.0792	1.19920	0.2348
$V_{(11)}$	-0.1665	0.0352	-4.7275	0.0000
$V_{(12)}$	0.11020	0.0453	2.43430	0.01770
$V_{(13)}$	-0.0930	0.0792	-1.1736	0.0248
$V_{(21)}$	0.02140	0.0352	0.60830	0.5451
$V_{(22)}$	-0.0836	0.0453	-1.8466	0.2694
$V_{(23)}$	0.03530	0.0792	0.44600	0.6571
$V_{(31)}$	0.00450	0.0317	0.14160	0.8878
$V_{(32)}$	-0.0324	0.0396	-0.8173	0.4167
$V_{(33)}$	0.08740	0.1585	0.55180	0.5830
$T_{(11)}$	-0.3859	0.1585	-2.4349	0.0176
$T_{(12)}$	0.59570	0.1585	3.75870	0.0004
$T_{(13)}$	0.02150	0.0453	0.47420	0.6369
$T_{(14)}$	0.02180	0.0352	0.61950	0.5378
$T_{(21)}$	-0.0181	0.1585	-0.1140	0.9096
$T_{(22)}$	-0.0709	0.1585	-0.4473	0.6562
$T_{(23)}$	0.10730	0.0453	2.36920	0.0208
$T_{(24)}$	-0.0412	0.0352	-1.1696	0.2464
$T_{(31)}$	1.10610	0.3170	3.48990	0.0009
$T_{(32)}$	-0.3620	0.3170	-1.1420	0.2576
$T_{(33)}$	0.37740	0.0453	8.33540	0.0000
$T_{(34)}$	-0.2638	0.0288	-9.1537	0.0000
$T_{(41)}$	-0.2216	0.3170	-0.6990	0.4870
$T_{(42)}$	-1.0884	0.3170	-3.4340	0.0010
$T_{(43)}$	0.29910	0.0528	5.66220	0.0000
$T_{(44)}$	-0.4182	0.0264	-15.833	0.0000
s_{max}	0.13270	0.0106	12.5598	0.0000

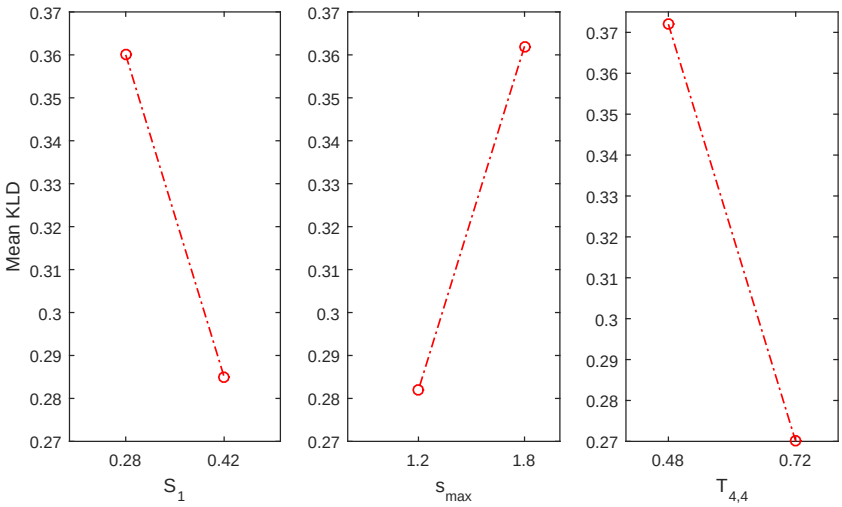


Figure E.3: Most sensitive parameters in the model.

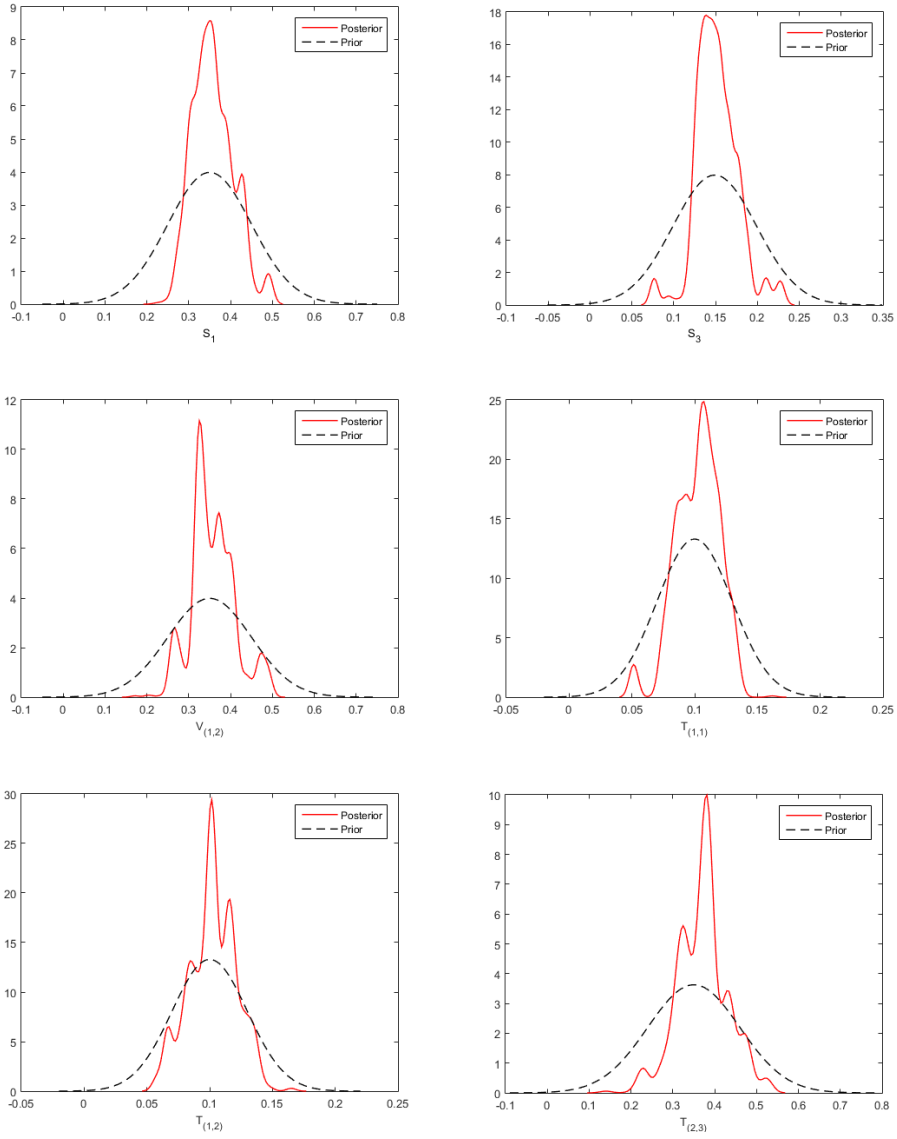


Figure E.4: Prior and Posterior distributions over parameters(Set 2).

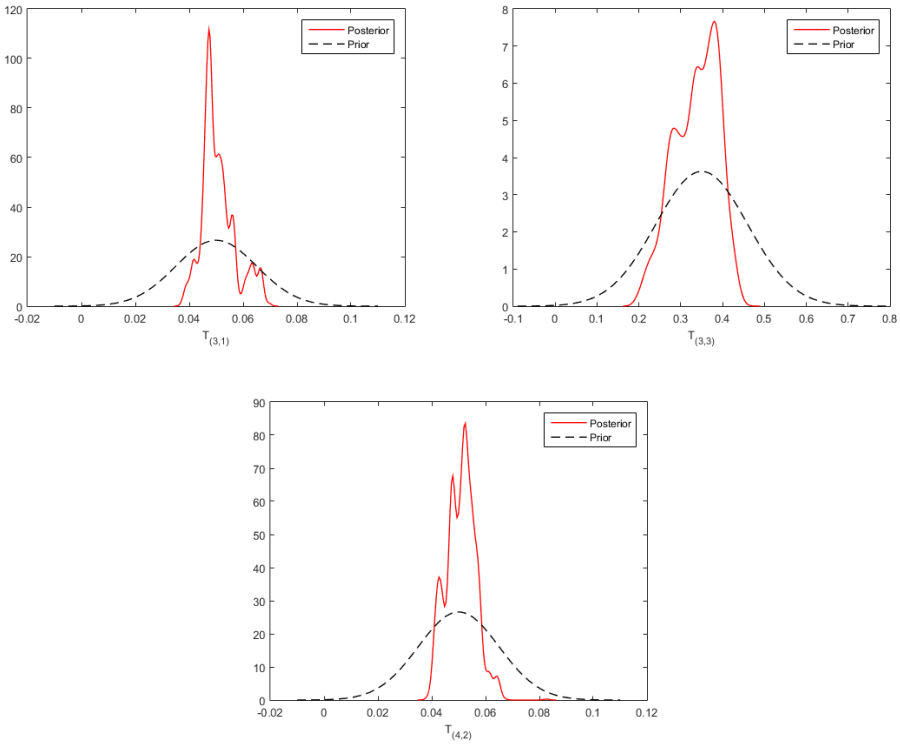


Figure E.5: Prior and Posterior distributions over parameters (Set 2).

Appendix F

Bearing Only SLAM Initialisation Methods

The main challenge with MonoSLAM algorithms is that the position of a feature is not globally observable from a single measurement. As a result, its initial position is poorly known. However, upon successive observations of the feature, the estimate of the feature position can be improved. Therefore, qualitatively, a feature can be considered to be in one of two conditions: *poorly-localised*, and *well-localised*. In the poorly localised condition, the position uncertainty is sufficiently large that a KF representation of Cartesian position performs poorly. In the well-localised condition, the feature uncertainty is sufficiently small that regular KF updates can be applied. The three algorithms we consider here are— DI, IDP and AHP — all utilise this basic structure, but the way in which they handle the poorly-localised condition is different.

F.1 Delayed Initialisation Predict and Update Model Jacobian Computation

$$\nabla \mathbf{F}_k = \begin{bmatrix} \frac{\partial \mathbf{p}_k^n}{\partial \mathbf{p}_{k-1}^n} & \frac{\partial \mathbf{p}_k^n}{\partial \mathbf{v}_{k-1}^n} & \frac{\partial \mathbf{p}_k^n}{\partial \mathbf{\Psi}_{k-1}} \\ \frac{\partial \mathbf{v}_k^n}{\partial \mathbf{p}_{k-1}^n} & \frac{\partial \mathbf{v}_k^n}{\partial \mathbf{v}_{k-1}^n} & \frac{\partial \mathbf{v}_k^n}{\partial \mathbf{\Psi}_{k-1}^n} \\ \frac{\partial \mathbf{\Psi}_k^n}{\partial \mathbf{p}_{k-1}^n} & \frac{\partial \mathbf{\Psi}_k^n}{\partial \mathbf{v}_{k-1}^n} & \frac{\partial \mathbf{\Psi}_k}{\partial \mathbf{\Psi}_k^n} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \triangle t\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \triangle t \frac{\partial \left(\mathbf{E}_{b,k-1}^n f_k^b \right)}{\partial \mathbf{\Psi}_{k-1}^n} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} + \triangle t \frac{\partial \left(\mathbf{E}_{b,k-1}^n \omega_k^b \right)}{\partial \mathbf{\Psi}_{k-1}^n} \end{bmatrix},$$

$$\nabla \mathbf{G}_k = \begin{bmatrix} \frac{\partial \mathbf{p}_k^n}{\partial f_k^b} & \frac{\partial \mathbf{p}_k^n}{\partial \omega_k^b} \\ \frac{\partial \mathbf{v}_k^n}{\partial f_k^n} & \frac{\partial \mathbf{v}_k^n}{\partial \omega_k^b} \\ \frac{\partial \mathbf{\Psi}_k^n}{\partial f_k^b} & \frac{\partial \mathbf{\Psi}_k^n}{\partial \omega_k^b} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{C}_{b,k-1}^n & \mathbf{0} \\ \mathbf{0} & \mathbf{E}_{k-1} \end{bmatrix}.$$

$$\nabla \mathbf{H}_{=} \begin{bmatrix} \frac{\partial \rho_k}{\partial \left(\mathbf{p}_{k-1}^n, \mathbf{v}_{k-1}^n, \mathbf{\Psi}_{k-1}^n, \mathbf{x}_{mi,k-1}^n \right)} \\ \frac{\partial \varphi_k}{\partial \left(\mathbf{p}_{k-1}^n, \mathbf{v}_{k-1}^n, \mathbf{\Psi}_{k-1}^n, \mathbf{x}_{mi,k-1}^n \right)} \\ \frac{\partial \theta_k}{\partial \left(\mathbf{p}_{k-1}^n, \mathbf{v}_{k-1}^n, \mathbf{\Psi}_{k-1}^n, \mathbf{x}_{mi,k-1}^n \right)} \end{bmatrix}. \tag{F.1}$$

F.2 Direction and Transformation Matrices

F.2.1 Body to Navigation Frame

In our work the navigation frame is rotated and fitted into the body frame in the sequences of yaw ψ, pitch θ and roll ϕ, hence the transformation matrix from the navigation to the body frame $\mathbf{C}_{n,}^{b}$ is constructed by multiplying the consecutive rotations matrices in the same sequences, as [227],

$$\mathbf{C}_{n,k}^{b} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\cos\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{C}_{b,k}^{n} = \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\theta\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\cos\psi & \text{-}\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ \text{-}\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}.$$

As the body frame rotates with respect to the navigation frame, the Euler angles also change according to the gyro measurement of the angular rate vector. Based on the sequence of rotations from the navigation to body frame, the angular rate of the body frame measured from the gyro is transformed into the Euler rate. Since the rotation in roll axis take place last, the roll rate $\dot{\phi}$ is equivalent to the angular rate along the x-axis of the body frame ω_x. The pitch rate $\dot{\theta}$ however is transformed according to the roll angle and yaw rate $\dot{\psi}$ is transformed according to the roll and pitch angles. The sum of these three Euler rates gives us the angular rates in the body frame [228].

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & \text{-}\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}.$$

The inverse of this equation gives us an expression for the rates of Euler angles

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \mathbf{E}_{b,k}^{n}\omega_{nb,k}^{b} = \begin{bmatrix} 1 & \sin\phi\sin\theta/\cos\theta & \cos\phi\sin\theta/\cos\theta \\ 0 & \cos\phi & \text{-}\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}.$$

In order to perform the transformation from navigation to body frame, the inverse or transpose of $\mathbf{C}^n_{b,}$ and $\mathbf{E}^n_{b,}$ are used, giving us $\mathbf{C}^b_{n,}$ and $\mathbf{E}^b_{n,}$.

F.2.2 Vision Sensor to Body Frame

The assumption made for our simulation is that simulated vision sensor is fixed at the centre of platform's body pointing downwards, so $\mathbf{C}^s_{b,}$ is the direction cosine matrix from body to sensor frame, essentially representing the relative transformation, which is an identity matrix in our case. In real world application however, this will not be the case as the camera will have some offset which would be measured by hand and also the coordinate frames of the two would not much exactly so a rotation matrix would be calculated to perform rotation between the body and camera coordinate frames and vice verse.

$$\mathbf{C}^s_{b,k} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \mathbf{C}^b_{s,k} = (\mathbf{C}^s_{b,k})^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Also,

$$\mathbf{P}^s_{sb} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

F.3 Inverse Depth Parameterisation

The intuition behind the IDP algorithm is that a poorly-localised landmark can be maintained and updated in a Kalman filter, but *not* using a Cartesian representation. Rather, the feature position is represented as a six dimensional vector [212] :

$$\mathbf{m}^n_{i,k|j} = \begin{bmatrix} \mathbf{p}^n_i \\ \varphi_i \\ \upsilon_i \\ \rho_i \end{bmatrix}_{k|j}.$$

which can be converted back to Cartesian by:

$$\mathbf{m}^n_i = \mathbf{p}^n_i + \mathbf{n}(\varphi_i, \upsilon_i)/\rho_i. \tag{F.2}$$

where $\mathbf{p}^n_i = [x^n_i y^n_i z^n_i]^T$ is the camera optical centre from where the 3D point was first observed, hence the anchor point, the ray directional vector for i^{th} feature $\mathbf{n}(\varphi_i, \upsilon_i)$ representing a vector

in terms of the direction of azimuth and elevation angles φ_i and υ_i respectively

$$\mathbf{n} = [\cos\varphi_i \sin\upsilon_i, -\sin\varphi_i, \cos\varphi_i \cos\upsilon_i]^T ,$$

and ρ is the inverse of the distance $\rho_i = 1/d_i$. The initial value of ρ_i is derived assuming that the distance, $d \geq d_{min}$. Therefore, ρ_i is initially chosen to have the mean and covariance so that it covers a range from infinity to a predefined close distance d_{min}.

In other words, the feature is encoded in terms of the epipolar line which it lies on (starting at $\mathbf{p}_i^n$ and with direction $\mathbf{n}(\varphi_i, \upsilon_i)$) together with the *inverse* of the distance from the start of the line to the point of the feature (ρ_i). The reason for this formulation is that once a feature has been initialised, the unit vector of the observation (computed from $\mathbf{z}_i^k$) can be computed from

$$\overline{\mathbf{u}}_{i,k}^s = \mathbf{K}\mathbf{C}_b^s \mathbf{C}_{n,k}^b \left[\rho_i \left(\mathbf{p}_i^n - \mathbf{p}_k^n\right) + \mathbf{n}(\varphi_i, \upsilon_i)\right] . \tag{F.3}$$

where $\mathbf{K}$ is the intrinsic camera calibration matrix and $\mathbf{p}_k^n$ is the current platform position at time k. In other words, the effect of the parametrisation is to turn the large uncertain range estimate into a multiplicative uncertainty, which is is better-behaved than a large divisible uncertainty.

F.3.1 Feature Initialisation

With this parametrisation, features can be initialised using only one captured image of the feature. When the feature is observed again, if at low parallax, the feature will be used simply to determine observing sensors orientation while keeping the feature depth uncertain, however if the sensor has translated enough to produce good parallax, then the feature depth estimation is improved.

When initialising, the initial location of the feature is calculated by function:

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ \hat{z}_i \\ \hat{\varphi}_i \\ \hat{\upsilon}_i \\ \hat{\rho}_i \end{bmatrix} = \mathbf{y}\left(\hat{\mathbf{p}}_k^n, \hat{\mathbf{\Psi}}_k^n, \mathbf{h}_{i,k}, \rho_{0,k}\right) , \tag{F.4}$$

where $\hat{\mathbf{\Psi}}_k^n$ is the estimated orientation of the platform at time k and $\mathbf{h}_{i,k}$ is the pixel coordinates

of the feature. Using this function, the projection ray point is set to the current sensor position

$$[\hat{x}_i\ \hat{y}_i\ \hat{z}_i]^T = \hat{\mathbf{p}}_k^n,$$

and the projection ray directional vector is calculated by:

$$\begin{pmatrix} \hat{\varphi}_i \\ \hat{\upsilon}_i \end{pmatrix} = \begin{pmatrix} \text{atan}\left(-\overline{\mathbf{u}}_y^n, \sqrt{\overline{\mathbf{u}}_x^{n2} + \overline{\mathbf{u}}_y^{n2}}\right) \\ \text{atan}\left(\frac{\overline{\mathbf{u}}_x^n}{\overline{\mathbf{u}}_y^n}\right) \end{pmatrix} \tag{F.5}$$

where

$$\overline{\mathbf{u}}_{i,k}^n = \mathbf{C}_{b,k}^n \mathbf{C}_s^b \mathbf{K}^{-1} \overline{\mathbf{u}}_{i,k}^s. \tag{F.6}$$

The covariance for $\hat{x}_i, \hat{y}_i, \hat{z}_i, \hat{\varphi}_i$ and $\hat{\upsilon}_i$ is derived from the image error covariance $\mathbf{R}_i$ and the state covariance estimate $\hat{\mathbf{P}}_k$ as:

$$\hat{\mathbf{P}}_k^{new} = \mathbf{J} \begin{pmatrix} \hat{\mathbf{P}}_k & 0 & 0 \\ 0 & \mathbf{R}_i & 0 \\ 0 & 0 & \sigma_\rho^2 \end{pmatrix} \mathbf{J}^T \tag{F.7}$$

where

$$\mathbf{J} = \begin{pmatrix} I & & 0 \\ \frac{\partial y}{\partial \mathbf{p}^n}, \frac{\partial y}{\partial \mathbf{\Psi}^n}, 0, \cdots, 0 & & \frac{\partial y}{\partial \mathbf{h}}, \frac{\partial y}{\partial \rho} \end{pmatrix}.$$

F.3.2 Feature Conversion to Cartesian Coordinate

For every feature in the map coded in inverse depth, after each estimation step, the feature estimate is checked if *well-localised* according to a linearity index L_d.

$$L_d = \frac{4\sigma_d}{d_i} |\cos\alpha| \tag{F.8}$$

where $\sigma_d = \frac{\sigma_\rho}{\rho_i^2}$, $\sigma_\rho = \sqrt{\mathbf{P}_{m_i m_i(6,6)}}$ and $\cos\alpha = \mathbf{n}\,[\hat{\mathbf{m}}_i^n - \hat{\mathbf{p}}_k^n] \parallel [\hat{\mathbf{m}}_i^n - \hat{\mathbf{p}}_k^n] \parallel^{-1}$.

The feature $\hat{\mathbf{m}}_i^n$ is computed using (F.2) and $\mathbf{P}_{m_i,m_i}$ is the sub-matrix 6×6 corresponding to the considered feature.

If the index L_d falls below a threshold, the feature is deemed to be *well-localised* or *Gaussian* distributed, and the inverse depth parametrisation is converted into Cartesian coordinates using (F.2) and the full state covariance matrix $\mathbf{P}$ is transformed with the corresponding Jacobian.

$$\mathbf{P}_{new} = \mathbf{J}\mathbf{P}\mathbf{J}^T, \mathbf{J} = \text{diag}\left(\mathbf{I}, \frac{\partial \mathbf{m}_i^n}{\partial \mathbf{y}_i}, \mathbf{I}\right). \tag{F.9}$$

Various improvements to this algorithm include using a common ray origin for multiple features defined within a single frame [229]. One issue is that such bearings-only algorithms have no sense of scale. This can be addressed through the use of inertial systems; in our case GPS works well. A more serious problem is that the algorithm can exhibit failure in which $\hat{\rho}_i$ can become negative [214,230]. Although various strategies can be used to circumvent these effects [214], they are symptomatic of the problem that the parametrisation can become unstable especially when the feature is far from the platform. In consequence of this limitation, Solà recently proposed an alternative parametrisation, the Anchored Homogeneous Point (AHP) [216].

F.4 Anchored Homogeneous Parametrisation

Solà argued that the formulation of the IDP in terms of azimuth and elevation angles introduces non-linear transformations into the filter which can contribute towards the exhibited non-linearities [216]. Therefore, he proposes to represent the poorly-localised map state by

$$\mathbf{m}^n_{i,k|j} = \begin{bmatrix} \mathbf{p}^n_i \\ u^n_i \\ v^n_i \\ w^n_i \\ \hat{\rho}_i \end{bmatrix}_{k|j} . \tag{F.10}$$

which could be converted back to Cartesian coordinate

$$\mathbf{m}^n_{i,k} = \mathbf{p}^n_i + \overline{\mathbf{u}}^n_{i,k}/\rho_i. \tag{F.11}$$

The crucial difference from (F.2) is that the vector which encodes the ray is *not* constrained to be a unit vector and that the optical direction is encoded with vector $\overline{\mathbf{u}}^n_{i,k} = (u^n_i, v^n_i, w^n_i)$ avoiding the need for non-linear transformation (F.3) and (F.5).

Using this formulation once again, once a feature has been initialised, the unit vector of the observation (computed from $\mathbf{z}^k_i$) can be computed from

$$\overline{\mathbf{u}}^s_{i,k} = \mathbf{K}\mathbf{C}^s_b\mathbf{C}^b_{n,k}\left[\rho_i\left(\mathbf{p}^n_i - \mathbf{p}^n_k\right) + \overline{\mathbf{u}}^n_{i,k}\right]. \tag{F.12}$$

F.4.1 Feature Initialisation

To initialise a new feature in AHP the following function is used

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ \hat{z}_i \\ \hat{u}_i \\ \hat{v}_i \\ \hat{w}_i \\ \hat{\rho}_i \end{bmatrix} = \mathbf{y}\left(\hat{\mathbf{p}}_k^n, \hat{\mathbf{\Psi}}_k^n, \mathbf{h}_{i,k}, \rho_{0,k}\right), \tag{F.13}$$

Using this function, the projection ray point is set to the current sensor position using (F.3.1) and the projection ray directional vector is calculated using (F.6) omitting (F.5) and reducing the non-linearities of the conversion to φ_i and υ_i.

The covariance for $\hat{x}_i, \hat{y}_i, \hat{z}_i, \hat{u}_i$, $\hat{v}_i$ and $\hat{w}_i$ is derived from the image error covariance $\mathbf{R}_i$ and the state covariance estimate $\hat{\mathbf{P}}_k$ similar to (F.7).

F.4.2 Feature Conversion to Cartesian

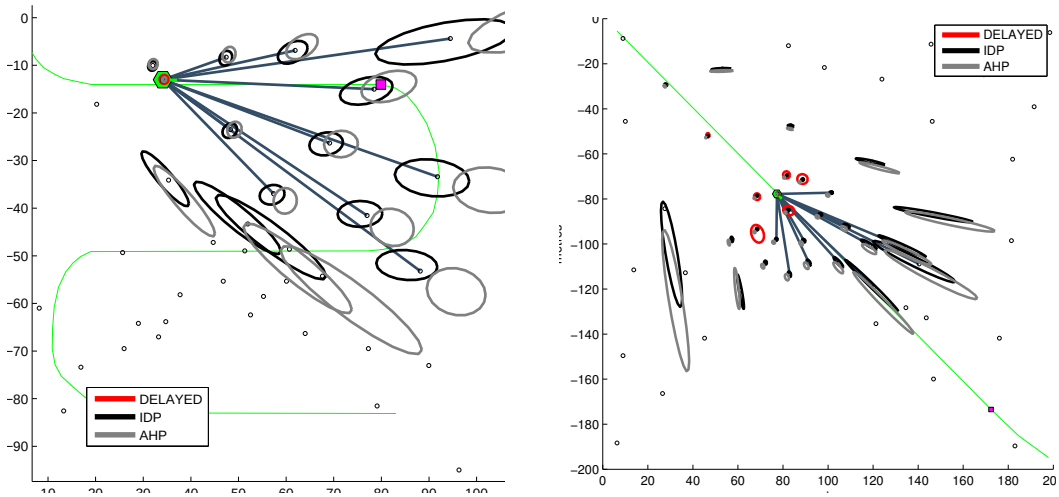
As in IDP, this representation is converted to Cartesian coordinates if the value of a linearity index falls below a critical value using functions in Section F.3.2. In this case, the conversion from Polar to Cartesian coordinates is carried out using (F.11).

Solà shows that the AHP provides significantly better performance than IDP for a scenario in which a ground platform drives between various features. This situation is very different from ours — in which a UAV flies over features and sees them from a considerable distance.

F.5 Bearing Only SLAM Experiment

To investigate the performance of the different Bearing only SLAM algorithms in localisation of point like features using UAVs, we perform a simulation study and compare their performance. In the simulations we treat both landmarks and evidence deposited by the LP as point features that are stationary.

Both delayed and undelayed bearing only SLAM initialisation methods come with certain drawbacks. The contribution of this experiment is to provide a detailed analysis of the three most prominent initialisation methods in bearing only SLAM and to compare the trade off of using one as oppose to the other for a particular scenario. Our experiments are performed using Monte Carlo simulations with simulated GPS and IMU data acquired of a quad-rotor model. To magnify the problems with each of the methods, we also investigated the effects of a range of conditions including target location, nadir angle of the camera and the trajectory of the UAV on the three bearings-only SLAM algorithms: DI, IDP, and AHP.



(a) S-trajectory **(b)** Straight line trajectory

Figure F.1: The trajectories (curves) used in both scenarios together with partially estimated maps from the different implementations. The actual landmark positions are represented by the small dots, and the estimated landmark positions for the different implementations are shown using their 3σ covariance ellipse.

F.5.1 Scenario

To analyse the performance of the different initialisation methods for a range of conditions, the simulations are based on two different scenarios illustrated in Figure F.1. The first is a square region whose sides are of length 200m and is populated with 50 landmarks. The trajectory is a straight line. This will provide the least parallax [212] especially if the target is located far along the line of the trajectory and is thus considered a "worst case". The second scenario consists of a square region whose sides are of length 100m and is populated with 40 landmarks. The trajectory is an S shape trajectory to provide high parallax [231]. Both trajectories are around 300 meters long.

Our simulations were designed to cover large trajectories of around 300 meters, which is around 1600 frames for S shaped trajectory and 1550 frames for straight line trajectory. The landmarks are configured so that the environment has areas consisting of sparse, populated and no landmarks to simulate real world scenario. Neither scenario includes loop closing because we do not anticipate that the quad-rotor will revisit part of the map. The quad-rotor itself mostly flies at a fixed altitude of 20m. We investigated two choices of the nadir angle (which affect $\mathbf{C}_b^s$): whether the camera points straight down at the ground, or whether it is oriented at 45° to the vertical (pointing in the direction of travel). The latter makes it possible for the quad-rotor to observe a larger part of the environment in any given frame, but trades off resolution to do so.

F.5.2 Comparison Criteria

The following criteria were used:

- **Estimation consistency.** Though Normalised Estimation Error Squared (NEES) used a lot in literature provides some evidence that estimates are covariance consistent, it provides no evidence of the actual consistency of features on 3D. When ground truth about a variable $\mathbf{x}_k$ is known, the NEES of its estimate is defined at each time k by

$$\epsilon_k = (\mathbf{x}_k - \hat{\mathbf{x}}_k)^T \mathbf{P}_k^{-1} (\mathbf{x}_k - \hat{\mathbf{x}}_k) .$$

  Under the hypothesis of consistent filtering of a linear Gaussian system, ϵ_k obeys a χ^2 distribution with $\dim(\mathbf{x}_k)$ degrees of freedom with expected ϵ_k converging to state dimension $\mathbf{E}[\epsilon_k] = \dim(x_k)$. Given N Monte Carlo runs, the average NEES is computed in

$$\overline{\epsilon}_k = \sum_{i=1}^{N} \epsilon_{ik}.$$

  If the average NEES is bellow some lower bound for some significant amount of time, the filter is conservative, and if above the upper bound, the filter is optimistic and therefore inconsistent.

- **Optimal Sub-Pattern Assignment (OSPA).** Used in multi-target tracking [232], it jointly considers cardinality and position errors and gives better representation of estimated map position errors or map consistency.

$$\overline{d}_p^{(c)}(\mathbf{X},\mathbf{Y}) = \left(\frac{1}{n}\left(\min_{\pi\in\Pi_n} \sum_{i=1}^{m} d^{(c)}\left(x_i, y_{\pi(i)}\right)^p + c^p(n-m)\right)\right)^{1/p}$$

  where $d^{(c)}(x,y) := \min(c,(x,y))$ is the distance between $x \in \mathbf{X} = \{x_i, x_2, \cdots, x_m\}, y \in \mathbf{Y} = \{y_1, y_2, \cdots, y_n\} \in W$ (bounded observation window) cut off at $c > 0$ and Π_k is the set permutations on $\{1, 2, \cdots, k\}$ for any $k \in N$. In our case $\mathbf{X}$ is the set of all well-localised landmarks and $\mathbf{Y}$ is the set of *all* landmarks detected by the platform so far. We used $p = 2$ and $c = 10m$ because these provide a good compromise between localisation and cardinality errors in this application.

- **Total number of features initialised.** This is the number of features which reach the well-localised state and are thus are "well-behaved".

- **Time to initialise.** This is average number of frames required until a feature becomes well-localised.

- **The baseline between the initial and last observation of landmarks.** This is the average change in angle required to estimate the landmark's location.
- **Computational complexity.** As complexity of SLAM is $O\left(n^3\right)$ we have stored the size of the state at each update cycle for each filter, representing the number of calculations required at each cycle.

F.5.3 Trajectories

In bearings-only tracking, the platform trajectory is critical to ensure observability and to obtain an accurate target localisation. An optimal course is to proceed at a fixed deviated angle, hence an optimal trajectory is a deviated pursuit curve [233]. To formalise this, Fawcett has proposed two simple rules [234]:

1. The platform has to move towards the target
2. The platform has to manoeuvre in order to maximise bearing rate (parallax)during the tracking and/or change of bearing rate between manoeuvres

Keeping above points in mind, Passerieux suggests that an optimal trajectory for a platform is composed of two or three "legs" with approximately equal lengths [231]. However undelayed initialisation claims it has the advantage to initialise objects well even in the absence of large parallax or objects along the optic axis further away, which would normally take long time for delayed initialisation.

F.5.4 Special Conditions

This section details the special conditions required to transform the observations from the platform controller to the filter in order to perform reliable estimation.

Our filter is developed to assume body coordinate in north, east, down (NED) frame, and the quad-rotor is based on coordinate frame north, west, up (NWU), so we have to convert the IMU data acquired from the sensor frame to the filter assumed body coordinate frame. To do this the IMU data is rotated 180°on the roll axes to have it converted to NED coordinate frame.

$$\left(\mathbf{C}_s^b\right) = \begin{bmatrix} \text{cos0cos0} & -\text{cos}\pi\text{sin0} + \text{sin0sin0} & \text{sin0sin0} + cos\pi\text{sin0} \\ \text{cos0sin0} & \text{cos}\pi\text{cos0} + \text{sin}\pi\text{sin0cos0} & -\text{sin}\pi\text{cos0} + \text{cos}\pi\text{sin0sin0} \\ \text{-sin0} & \text{sin}\pi\text{cos0} & \text{cos}\pi\text{cos0} \end{bmatrix},$$

$$\left(\mathbf{C}_s^b\right) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix},$$

Threshold	Baseline	Performance
$0\,^\circ$	$40\,^\circ$	Many features are detected and well localised, but tremendously expensive
$11.4\,^\circ$	$40\,^\circ$	Many detected features are detected and well localised, and computationally less expensive
$22.9\,^\circ$	$40\,^\circ$	Very similar to $11.4\,^\circ$ threshold
$28.6\,^\circ$	$40\,^\circ$	Faster and computationally less expensive but fewer number of landmarks initialised
$17.1\,^\circ$	$34.5\,^\circ$	Small number of landmarks initialised and with conservative observation NEES

Table F.1: Effect of threshold and baseline angles on the delayed initialisation method based on simulation performed for S shape trajectory with the camera pointing down.

$$\left(\mathbf{E}_s^b\right) = \begin{bmatrix} 1 & \sin\pi\sin 0/\cos 0 & \cos\pi\sin 0/\cos 0 \\ 0 & \cos\pi & \text{-}\sin\pi \\ 0 & \sin\pi/\cos 0 & \cos\pi/\cos 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix},$$

$$\boldsymbol{\omega}^b = \mathbf{E}_s^b \boldsymbol{\omega}^s,$$

$$\mathbf{f}^b = \mathbf{C}_s^b \mathbf{f}^s,$$

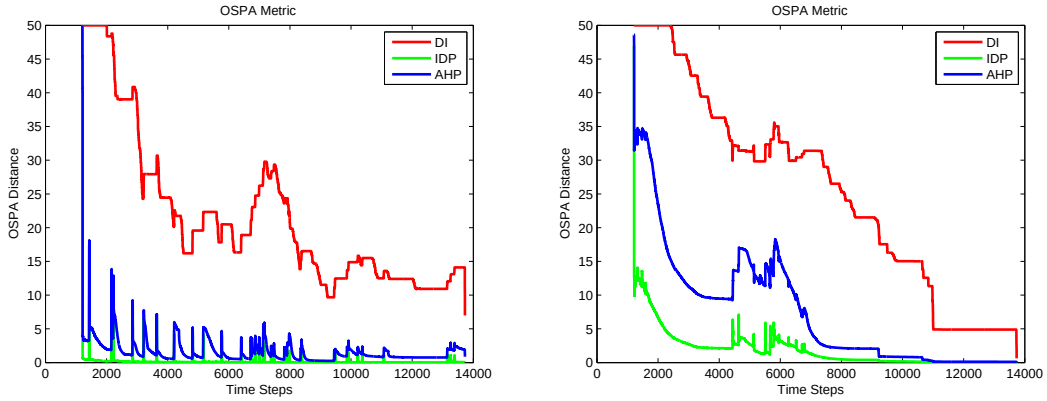
where $\boldsymbol{\omega}^b$ and $\mathbf{f}^b$ are the gyro and acceleration rates in filter assumed body coordinate frame, $\boldsymbol{\omega}^s$ and $\mathbf{f}^s$ are the gyro and accelerations rates in the sensor coordinate frame and $\mathbf{E}_s^b$ and $\mathbf{C}_s^b$ are the transformation matrix and direction cosine matrix used to convert the measurements from the sensor frame s to filter assumed body frame b.

F.5.5 Results and Discussion

Table F.2 summarises the performance results for the different algorithms and Figures F.2 and F.3 illustrate the plots for OSPA distance metrics and maximum Eigenvalue respectively for each filter. All results were computed for 20 Monte Carlo runs. Since we found that, qualitatively, the performance and issues for DI are different from those of IDP and AHP, we discuss DI and the IDP/AHP algorithms separately.

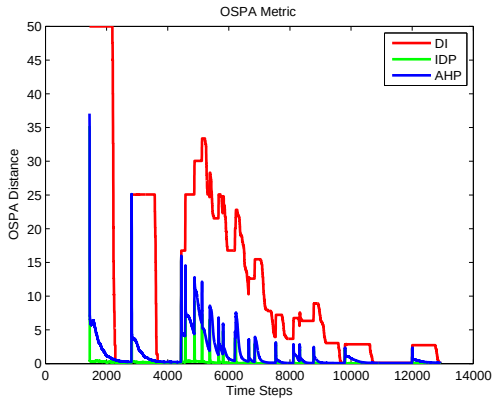
F.5.5.1 Delayed Initialisation Algorithm

From our simulated runs of the filter it is obvious that the DI algorithm produces very consistent performance across all the scenarios and sensor values tried. It produced consistent estimates as it waits for a good baseline to triangulate location of landmarks illustrated in Figure F.3, hence with least localisation error, and not exhibiting any sign of catastrophic failures in any of the runs. This is backed by the plots in Figure F.2 and Figure F.3, where it is clearly shown that DI has a smooth curve showing low uncertainty in landmarks localisation at initialisation instants, proving that this method produces reliable and well estimated maps of search area. However, we observed several issues which are all caused by the large baseline required by this method. As shown in Table F.2, the large baseline means that the DI algorithm takes an average

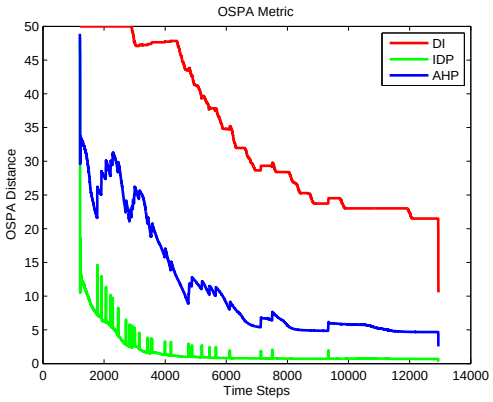


(a) S trajectory, nadir=0°.

(b) S trajectory, nadir=45°.



(c) Straight trajectory, nadir=0°.

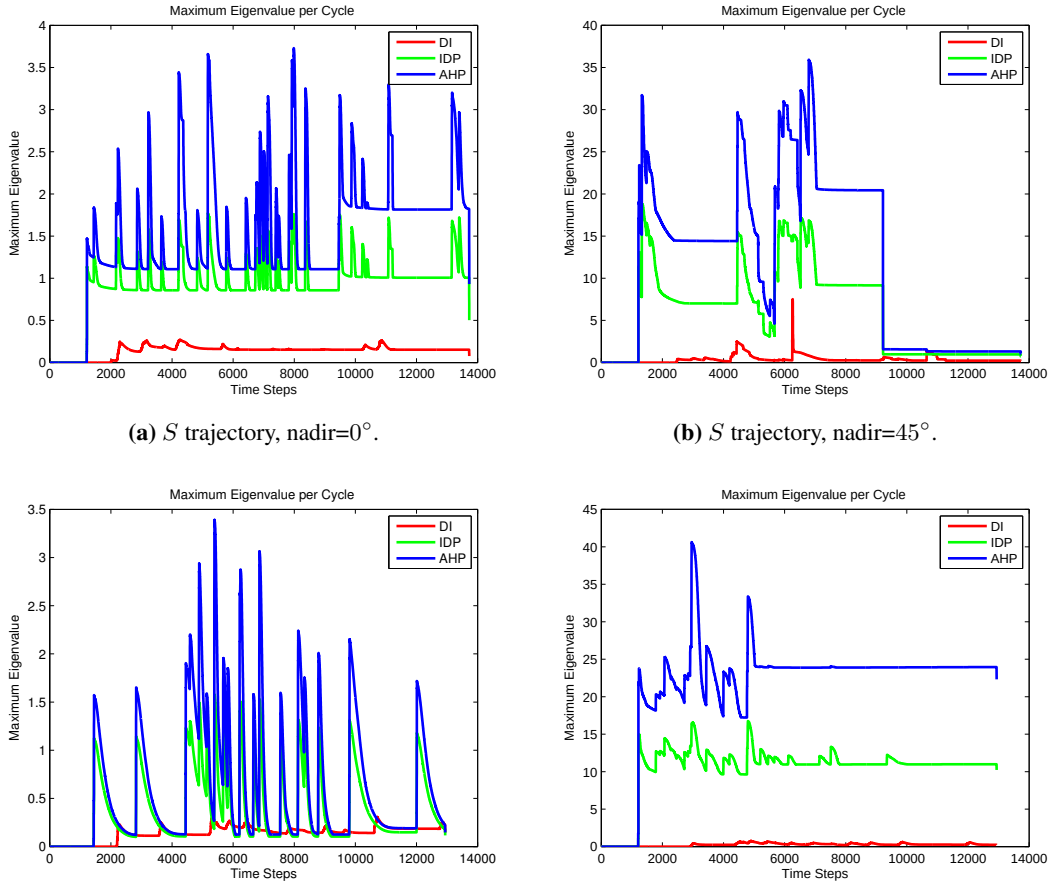


(d) Straight trajectory, nadir=45°.

Figure F.2: Time histories of OSPA Metric. As could be observed the DI OSPA distance is mainly due to cardinality, as it holds many observations of landmarks before they are initialised, hence the sharp drops of the plot at initialisation instants. But the OSPA distance for IDP and AHP are due to localisation error, as both of the methods hold only one observation of the landmarks, that are initialised immediately, and the peaks and distances are as a result of high uncertainty in observed landmarks, which reduces with re-observations, hence the curves

of more than 770 frames, which is over 150s of simulated time. This has obvious implications in terms of both the computational and storage costs of the filter (due to augmenting the poses of the platform at each observation of landmarks (8.18)). Furthermore, many landmarks are not initialised due to the trajectories chosen. In the worst case (straight line, camera at 45°) less than two thirds of the landmarks are actually initialised as shown in Table F.2.

To consider these effects, we investigated two changes to the algorithm. First, we reduced the baseline angle used to declare features well-localised. Second, we modified the augmentation logic so that the state is augmented and the observation is stored only if there is some minimal baseline between successive observations , The results of which is given in Table F.1. The affect of minimal baseline was observed when we run the system using a minimal baseline



(a) S trajectory, nadir=0°.

(b) S trajectory, nadir=45°.

(c) Straight trajectory, nadir=0°.

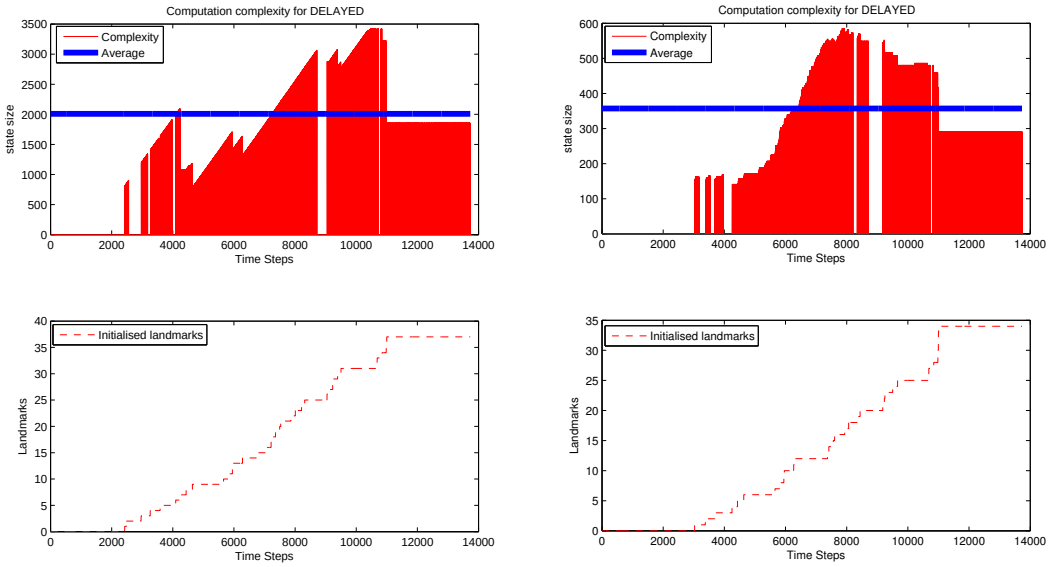
(d) Straight trajectory, nadir=45°.

Figure F.3: These figures illustrate the maximum of landmark Eigenvalues per cycle. As it could be observed nearly in all cases AHP has higher uncertainty compare to both IDP and Delayed methods. The spikes indicate the instants when a new landmark is observed and then with IDP and AHP they are instantly initialised hence high uncertainty initially and then sudden reduction in uncertainty with re-observation of the landmark, while in Delayed case, the landmark is initialised only when a good estimate of the depth is acquired, hence the smooth and low uncertainty.

of 11.4° and 0° Figure F.4, with baseline 40°, giving average state size of around 350 and 2000 respectively, which obviously deeply affects the performance, giving a factor of 27 difference in speed with the $O(N^3)$ scaling. However it was demonstrated that DI is able to both initialise landmarks and stay stable even in the absence of immediate utilisation of bearing information. In all scenarios DI was able to produce good results.

F.5.5.2 IDP and AHP

In all the scenarios, both IDP and AHP algorithms suffered from periodic negative depth problems, indicating that the filters had failed. We observed that this predominantly happened in two cases. The first case arose when the UAV observed distant landmarks. Because of the camera angle and the relatively low speed of the UAV, there is little parallax and observation



(a) Threshold = 0° **(b)** Threshold = 11.4°

Figure F.4: The impact of using the angle threshold. The computational cost is greatly reduced, with minimal change in the number of landmarks initialised and the time when initialisation occurs, proving it to be very important when dealing with large number of features or features that require long time to meet the baseline i.e when flying in a straight trajectory. In this figure, the top two plots illustrate the size of the filter state at each cycle (when the filter performs update) where the drops represent reduction in the state size when a new feature is initialisation and all the observations associated to it are removed [1]. The bottom figures however show the gradually incrementing number of landmarks that gets initialised.

noise terms dominated. The second case arose when the platform observed a landmark with high uncertainty from a number of very different angles. These difficulties were exacerbated when the vantage point to the nadir was 45°.

To overcome these difficulties, we found it was necessary to use truncated second order filters. Furthermore, we had to modify the parametrisation so that the state uses the *log* of the inverse of the depth [214]. However, we did not observe significant improvements when iterated Kalman filters (using either linearisation or second order terms) were used.

With these changes, we found that the IDP gave significantly better performance and, indeed, its performance is better than the DI algorithm. As Table F.2 shows, all the landmarks were initialised, and the number of frames required to get well localised landmarks were at least an order of magnitude less than that required for DI. However, we found that the AHP performed extremely poor in almost all circumstances with high OSPA distances shown in Figure F.2 where peaks are as a result of very uncertain landmark estimates illustrated in Figure F.3[2].

[2]The max Eigenvalue for IDP and AHP remain constant towards the end of scenarios (straight trajectory with

Parameters	S shaped with Camera Down			S shaped With Camera at 45°			Straight with camera down			Straight with camera at 45°		
Method	AHP	IDP	Delayed	AHP	IDP	Delayed	AHP	IDP	Delayed	AHP	IDP	Delayed
Cost/update	64	64	215	101	101	338	41	41	76	87	87	433
Landmarks	37	37	27	40	40	34	19	19	18	33	33	20
Frames	33	42	711	41	94	771	25	43	693	45	118	630
Baseline	3°	7.2°	46.5°	11.5°	18.9°	69°	1.18°	1.3°	47.5°	0.45°	0.74°	47°
Performance	OK	V.Good	V.Good	OK	V.Good	V.Good	OK	V.Good	Good	Bad	Good	Good

Table F.2: Average MC results (with performance based on state NEES and OSPA metric).

Since it is possible that this could be caused by an inaccurate initial depth estimate, we modified the initialisation of depth to exploit the fact that the landmark features lie close to the ground. However, we found that even with these accurate depth estimates, the qualitative performance did not change significantly. We found that if we further limited the maximum range that a landmark could lie at would be 60m, then consistent behaviour could be obtained. However, this is at a cost of greatly reducing the utility of the algorithm, and suggests that AHP is extremely poor at dealing with distant targets and little motion parallax.

F.6 Summary

Three different well established Monocular SLAM approaches were analysed and tested in terms of feature initialisation F. Our results show that the behaviour of these SLAM algorithms differ greatly from those published before. From an algorithmic perspective, we find that second order filters have a significant impact upon performance, but iterated forms of the filters do not.

We find that the AHP gives inferior response in all cases, and is not recommended. DI gives the most robust performance, but at the cost of significant latency in initialising the features. Furthermore, a sizeable fraction of features are never fully initialised. The IDP appears to give the best performance overall both in terms of immediate availability of landmarks and estimation accuracy. However, this robustness can only be achieved using a log parametrisation of depth.

nadir=45° and S trajectory shaped with nadir=0°) does not change as there are landmarks detected earlier that are not re-observed.

Bibliography

[1] H. K. Cordell, C. J. Betz, G. T. Green, *et al.*, "Nature-based outdoor recreation trends and wilderness," *International Journal of Wilderness*, vol. 14, no. 2, pp. 7–13, 2008.

[2] T. W. Heggie and M. E. Amundson, "Dead men walking: search and rescue in US National Parks," *Wilderness & Environmental Medicine*, vol. 20, no. 3, pp. 244–249, 2009.

[3] R. J. Koester, *Lost Person Behavior*. dbS Productions, 2008.

[4] D. Ferguson, "GIS for Wilderness search and rescue," *ESRI Federal User Conference*, 2008.

[5] J. A. Adams, C. M. Humphrey, M. A. Goodrich, J. L. Cooper, B. S. Morse, C. Engh, and N. Rasmussen, "Cognitive task analysis for developing unmanned aerial vehicle wilderness search support," *Journal of cognitive engineering and decision making*, vol. 3, no. 1, pp. 1–26, 2009.

[6] M. K. Alan Washburn, *Combat Modeling*, vol. 134 of *International Series in Operations Research & Management Science*. Springer US, 2009. pp 185-210.

[7] T. M. Express, "Drones to the rescue," 03 2014.

[8] BBC, "Drone operator explains how he found missing man." News, July 2014. http://www.bbc.com/news/technology-28423252.

[9] WMTV, "Family uses drone to help locate missing man." nbc.com, July 2014. http://www.nbc15.com/home/headlines/Fitchburg-Police-looking-for-a-missing-82-year-old-man-267433271.html.

[10] J. A. Adams, J. L. Cooper, M. A. Goodrich, C. Humphrey, M. Quigley, B. G. Buss, and B. S. Morse, "Camera-equipped mini uavs for wilderness search support: Task analysis and lessons from field trials," *Journal of Field Robotics*, vol. 25, no. 1-2, 2007.

[11] S. Hansen, T. McLain, and M. Goodrich, "Probabilistic searching using a small unmanned aerial vehicle," *Proceedings of AIAA Infotech@ Aerospace*, pp. 7–10, 2007.

[12] M.A.Goodrich, B. S.Morse, D.Gerhardt, J. L.Cooper, M.Quigley, J. A.Adams, and C.Humphrey, "Supporting wilderness search and rescue using a camera-equipped mini UAV: Research Articles," *J. Field Robot.*, vol. 25, no. 1-2, pp. 89–110, 2008.

[13] M. Quigley, B. Barber, S. Griffiths, and M. A. Goodrich, "Towards real-world searching with fixed-wing mini-UAVs," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 3028–3033, IEEE, 2005.

[14] M. A. Goodrich, J. L. Cooper, J. A. Adams, C. Humphrey, R. Zeeman, and B. G. Buss, "Using a mini-uav to support wilderness search and rescue: Practices for human-robot teaming," in *2007 IEEE International Workshop on Safety, Security and Rescue Robotics*, pp. 1–6, IEEE, 2007.

[15] J. Cooper and M. A. Goodrich, "Towards combining uav and sensor operator roles in uav-enabled visual search," in *Human-Robot Interaction (HRI), 2008 3rd ACM/IEEE International Conference on*, pp. 351–358, IEEE, 2008.

[16] F. Bourgault, T. Furukawa, and H. F. Durrant-Whyte, "Coordinated decentralized search for a lost target in a bayesian world," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 1, pp. 48–53, IEEE, 2003.

[17] S. Waharte, A. Symington, and N.Trigoni, "Probabilistic Search with Agile UAVs," in *Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA2010)*, (Anchorage, AK, USA), pp. 2840–2845, 3–8 May 2010.

[18] A. Symington, S. Waharte, S. J., and N.Trigoni, "Probabilistic Target Detection by Camera-Equipped UAVs," *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 4076–4081, May 3-8 2010.

[19] T. Chung and J. Burdick, "Multi-agent Probabilistic Search in a Sequential Decision-Theoretic Framework," *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 146 – 151, 2008.

[20] T.H.Chung and J. Burdick, "A Decision-Making Framework for Control Strategies in Probabilistic Search," *Robotics and Automation, 2007 IEEE International Conference on*, pp. 4386 – 4393, 2007.

[21] H. Bendea, P. Boccardo, S. Dequal, F. Giulio Tonolo, D. Marenchino, and M. Piras, "Low cost uav for post-disaster assessment," in *Proceedings of The XXI Congress of the International Society for Photogrammetry and Remote Sensing, Beijing (China), 3-11 July 2008*, 2008.

[22] S. Cameron, S. Hailes, S. Julier, S. McClean, G. Parr, N. Trigoni, M. Ahmed, G. McPhillips, R. De Nardi, J. Nie, *et al.*, "Suaave: Combining aerial robots and wireless networking," in *25th Bristol International UAV Systems Conference*, pp. 1–14, 2010.

[23] H. Flynn and S. Cameron, "Multi-modal people detection from aerial video," in *Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013*, pp. 815–824, Springer, 2013.

[24] H. Wang and S. Julier, "Path planning in partially known environments," 2011.

[25] T. Patterson, S. McClean, G. Parr, P. Morrow, L. Teacy, and J. Nie, "Integration of Terrain Image Sensing with UAV Safety Management Protocols," *Sensor Systems and Software*, pp. 36–51, 2011.

[26] T. Patterson, S. McClean, P. Morrow, and G. Parr, "Utilizing Geographic Information System Data for Unmanned Aerial Vehicle Position Estimation," in *Computer and Robot Vision (CRV), 2011 Canadian Conference on*, pp. 8–15, IEEE, 2011.

[27] S. I. McClean, B. W. Scotney, T. Patterson, P. J. Morrow, and G. Parr, "Fusion of data and knowledge for safe uav landing," *International Journal of Software and Informatics*, no. 6, pp. pp. 381–398, 2012.

[28] T. Patterson, *Fusing Knowledge and Image Sensor Data for Mission Critical Control in Unmanned Aerial Vehicles*. PhD thesis, Computing and Information Engeineering, Faculty of Engineering of the University of Ulster, February 2013.

[29] R. De Nardi, "The QRSim Quadrotors Simulator," *RN*, vol. 13, no. 08, p. 08, 2013.

[30] "Ascending Technologies Hummingbird." http://www.asctec.de. Last accessed 6th April, 2010.

[31] S. Sukkarieh, E. Nettleton, J.-H. Kim, M. Ridley, A. Goktogan, and H. Durrant-Whyte, "The anser project: Data fusion across multiple uninhabited air vehicles," *The International Journal of Robotics Research*, vol. 22, no. 7-8, pp. 505–539, 2003.

[32] R. J. Koester, "International Search and Rescue Incident Database (ISRID)." http://www.dbs-sar.com/. Last accessed 05/10/2014.

[33] D. C. Cooper and J. R. Frost, "Compatibility of Land SAR Procedures with Search Theory," 2003.

[34] P. A. Dudchenko, *Why people get lost: the psychology and neuroscience of spatial cognition*. Oxford University Press, 2010.

[35] A. Macwan, G. Nejat, and B. Benhabib, "Target-motion prediction for robotic search and rescue in wilderness environments," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 41, no. 5, pp. 1287–1298, 2011.

[36] C. Twardy, R. Koester, and R. Gatt, "Missing person behaviour: an australian study," *Final Report to the Australian National SAR Council*, vol. 72, 2006.

[37] E. H. Cornell, C. D. Heth, and D. M. Alberts, "Place recognition and way finding by children and adults," *Memory & Cognition*, vol. 22, no. 6, pp. 633–643, 1994.

[38] C. Donald Heth, E. H. Cornell, and T. L. Flood, "Self-ratings of sense of direction and route reversal performance," *Applied cognitive psychology*, vol. 16, no. 3, pp. 309–324, 2002.

[39] T. S. Castle, "Coordinated inland area search and rescue (sar) planning and execution tool.," tech. rep., DTIC Document, 1998.

[40] J. Doke, *Analysis of search incidents and lost person behavior in Yosemite National Park*. PhD thesis, University of Kansas, 2012.

[41] D. Perkins and P. Roberts, "The u.k. missing person behaviour study," *The center for search research*, August 2005.

[42] F. Bourgault, T. Furukawa, and H. F. Durrant-Whyte, "Optimal search for a lost target in a bayesian world," in *Field and service robotics*, pp. 209–222, Springer, 2003.

[43] T. Furukawa, F. Bourgault, B. Lavis, and H. F. Durrant-Whyte, "Recursive Bayesian Search-and-Tracking Using Coordinated UAVs for Lost Targets," *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 2521–2526, 05 2006.

[44] T.H.Chung, M. Kress, and J.O.Royset, "Probabilistic search optimization and mission assignment for heterogeneous autonomous agents," *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pp. 939 – 945, 2009.

[45] Z. Chen, "Bayesian filtering: From kalman filters to particle filters, and beyond," *Statistics*, vol. 182, no. 1, pp. 1–69, 2003.

[46] F. Bourgault and H. Durrant-Whyte, "Communication in general decentralized filters and the coordinated search strategy," in *Proc. of The 7th Int. Conf. on Information Fusion*, pp. 723–730, Citeseer, 2004.

[47] F. Bourgault, T.Furukawa, and H. F. Durrant-Whyte, "Process Model, Constraints, and the Coordinated Search Strategy," *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 5, pp. 5256–5261, 26 April – 1 May 2004.

[48] E.-M. Wong, F. Bourgault, and T. Furukawa, "Multi-vehicle bayesian search for multiple lost targets," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 3169–3174, IEEE, 2005.

[49] J. A. S. William Syrotuck, *Analysis of Lost Person Behavior: An aid to search planning*. Westmoreland N.Y.: Arner Publications, 1977.

[50] R. W. Beard, D. Kingston, M. Quigley, D. Snyder, R. Christiansen, W. Johnson, T. McLain, and M. Goodrich, "Autonomous vehicle technologies for small fixed-wing uavs," *Journal of Aerospace Computing, Information, and Communication*, vol. 2, no. 1, pp. 92–108, 2005.

[51] M. Quigley, B. Barber, S. Griffiths, and M. A. Goodrich, "Towards real-world searching with fixed-wing mini-uavs," tech. rep., DTIC Document, 2005.

[52] J. Li and X.-x. Sun, "A route planning's method for unmanned aerial vehicles based on improved a-star algorithm [j]," *Acta Armamentarii*, vol. 7, pp. 788–792, 2008.

[53] P. O. Pettersson and P. Doherty, "Probabilistic roadmap based path planning for an autonomous unmanned helicopter," *Journal of Intelligent & Fuzzy Systems*, vol. 17, no. 4, pp. 395–405, 2006.

[54] L. Blackmore, H. Li, and B. Williams, "A probabilistic approach to optimal robust path planning with obstacles," in *American Control Conference, 2006*, pp. 7–pp, IEEE, 2006.

[55] D. Bertsekas and S. Shreve, "Stochastic optimal control: The discrete time case," *Athena Scientific, Belmont*, 1978.

[56] K. Trummel and J. Weisinger, "Technical note: The complexity of the optimal searcher path problem," *Operations Research*, vol. 34, no. 2, pp. 324–327, 1986.

[57] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[58] L. Lin and M. A. Goodrich, "Uav intelligent path planning for wilderness search and rescue," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 709–714, IEEE, 2009.

[59] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[60] S. Waharte, N. Trigoni, and S. J. Julier, "Coordinated search with a swarm of uavs," in *Sensor, Mesh and Ad Hoc Communications and Networks Workshops, 2009. SECON Workshops' 09. 6th Annual IEEE Communications Society Conference on*, pp. 1–3, IEEE, 2009.

[61] L. Lin and M. A. Goodrich, "A Bayesian approach to modeling lost person behaviors based on terrain features in Wilderness Search and Rescue," *Proceedings of the 18th Conference on Behavior Representation in Modeling and Simulation, Sundance, UT, 31 March-2 April 2009*, vol. 16, pp. 300–323, 2009.

[62] T. J. Pingel, "Modeling slope as a contributor to route selection in mountainous areas," *Cartography and Geographic Information Science*, vol. 37, no. 2, pp. 137–148, 2010.

[63] D. R. Proffitt, M. Bhalla, R. Gossweiler, and J. Midgett, "Perceiving geographical slant," *Psychonomic Bulletin & Review*, vol. 2, no. 4, pp. 409–428, 1995.

[64] "Hikers." Google Images.

[65] "Landmap." http://www.landmap.ac.uk/. Last accessed 22nd February, 2011.

[66] "Getmapping." http://www2.getmapping.com/Home. Last accessed 30/05/2013.

[67] "Ordnance Survey." http://www.ordnancesurvey.co.uk/oswebsite/opendata/. Last accessed 22nd February, 2011.

[68] "Bluesky." http://www.bluesky-world.com/. Last accessed 20 October 2013.

[69] S. Chib and E. Greenberg, "Understanding the metropolis-hastings algorithm," *The american statistician*, vol. 49, no. 4, pp. 327–335, 1995.

[70] M. Raubal, "Human wayfinding in unfamiliar buildings: a simulation with a cognizing agent," *Cognitive Processing*, vol. 2, no. 3, pp. 363–388, 2001.

[71] J. Weisman, "Evaluating Architectural Legibility Way-Finding in the Built Environment," *Environment and behavior*, vol. 13, no. 2, pp. 189–204, 1981.

[72] A. Turner and A. Penn, "Evolving direct perception models of human behavior in building systems," in *Pedestrian and Evacuation Dynamics 2005*, pp. 411–422, Springer, 2007.

[73] M. Bierlaire, G. Antonini, and M. Weber, "Behavioral dynamics for pedestrians," in *Moving Through Nets: The Physical and Social Dimensions of Travel. 10th International Conference on Travel Behavior Research. Lucerne*, 2003.

[74] K. Hill, *The Psychology of Lost. In Lost Person Behaviour*. National SAR Secretariat, Ottawa, Canada, 1998.

[75] J. Peponis, C. Zimring, and Y. K. Choi, "Finding the building in wayfinding," *Environment and behavior*, vol. 22, no. 5, pp. 555–590, 1990.

[76] R. C. Dalton, "The Secret Is To Follow Your Nose Route Path Selection and Angularity," *Environment and Behavior*, vol. 35, no. 1, pp. 107–131, 2003.

[77] C. J. E. Castle and A. T.Crooks, "Principles and Concepts of Agent-Based Modelling for Developing Geospatial Simulations," *Centre for Advanced Spatial Analysis (UCL), UCL (University College London), Centre for Advanced Spatial Analysis (UCL),*, pp. 1 – 60, 2006.

[78] Wikipedia, "Agent-based model." https://en.wikipedia.org/wiki/Agent-based-model, 08 2016. 04.

[79] P. P. González, M. Cárdenas, D. Camacho, A. Franyuti, O. Rosas, and J. Lagúnez-Otero, "Cellulat: an agent-based intracellular signalling model," *BioSystems*, vol. 68, no. 2, pp. 171–185, 2003.

[80] M. d'Inverno and R. Saunders, "Agent-based modelling of stem cell self-organisation in a niche," in *Engineering Self-Organising Systems*, pp. 52–68, Springer, 2005.

[81] T. Emonet, C. M. Macal, M. J. North, C. E. Wickersham, and P. Cluzel, "Agentcell: a digital single-cell assay for bacterial chemotaxis," *Bioinformatics*, vol. 21, no. 11, pp. 2714–2721, 2005.

[82] N. Gilbert and K. Troitzsch, *Simulation for the social scientist*. McGraw-Hill International, 2005.

[83] C. Gloor, D. Cavens, E. Lange, K. Nagel, and W. A. Schmid, "A Pedestrian Simulation for Very Large Scale Applications," *Multi-Agenten-Systeme in der Geographie*, 2003.

[84] L. Tesfatsion, "Agent-based computational economics: Growing economies from the bottom up," *Artificial life*, vol. 8, no. 1, pp. 55–82, 2002.

[85] L. Tesfatsion, "Agent-based computational economics," *Scholarpedia*, vol. 2, no. 2, p. 1970, 2007.

[86] T. A. Kohler, G. J. Gumerman, and R. G. Reynolds, "Simulating ancient societies," *Scientific American*, vol. 293, no. 1, pp. 76–84, 2005.

[87] J. H. Christiansen and M. R. Altaweel, "Understanding ancient societies: A new approach using agent-based holistic modeling," *Structure and Dynamics*, vol. 1, no. 2, 2005.

[88] W. K. V. Chan, Y.-J. Son, and C. M. Macal, "Agent-based simulation tutorial-simulation of emergent behavior and differences between agent-based simulation and discrete-event simulation," in *Proceedings of the Winter Simulation Conference*, pp. 135–150, Winter Simulation Conference, 2010.

[89] T. W. Crawford, J. P. Messina, S. M. Manson, and D. O'Sullivan, "Complexity science, complex systems, and land-use research," *Environment and Planning B: Planning and Design*, vol. 32, no. 6, pp. 792–798, 2005.

[90] L. An, "Modeling human decisions in coupled human and natural systems: review of agent-based models," *Ecological Modelling*, vol. 229, pp. 25–36, 2012.

[91] Wikipedia, "Emergence." http://en.wikipedia.org/wiki/Emergence, February 2015.

[92] C. M. Macal and M. J. North, "Tutorial on agent-based modeling and simulation," in *Proceedings of the 37th conference on Winter simulation*, pp. 2–15, Winter Simulation Conference, 2005.

[93] C. Gloor, L. Mauron, and K. Nagel, "A pedestrian simulation for hiking in the Alps," in *In proceedings of Swiss transport conference (STRC), Monte Verita*, Citeseer, 2003.

[94] N.Ronald, L. Sterling, and M. Kirley, "An agent-based approach to modelling pedestrian behaviour," *International Journal of simulation Systems, Science and Technology*, vol. 8, p. 25039, 2007.

[95] K. Teknomo, "Application of microscopic pedestrian simulation model," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 9, no. 1, pp. 15–27, 2006.

[96] F. Klügl and G. Rindsfüser, *Large-scale agent-based pedestrian simulation*. Springer, 2007.

[97] T. Karmakharm, P. Richmond, and D. M. Romano, "Agent-based large scale simulation of pedestrians with adaptive realistic navigation vector fields.," *TPCG*, vol. 10, pp. 67–74, 2010.

[98] E. Bonabeau, "Agent-based modeling Methods and techniques for simulating human systems," *Proceedings of the National Academy of Sciences*, vol. 99, pp. 7280–7287, 2002.

[99] H.R.Gimblett, "Modelling Human-Landscape Interactions in Spatially Complex Settings Where are we and where are we going?," *Paper presented at the MODSIM 2005, International Congress on Modelling and Simulation. Modelling and Simulation Society of Australia and New Zealand*, pp. 11–20, 2005.

[100] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *Signal Processing, IEEE Transactions on*, vol. 50, no. 2, pp. 174–188, 2002.

[101] N. J. Gordon, D. J. Salmond, and A. F. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," in *IEE Proceedings F-Radar and Signal Processing*, vol. 140, pp. 107–113, IET, 1993.

[102] K. Kanazawa, D. Koller, and S. Russell, "Stochastic simulation algorithms for dynamic probabilistic networks," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pp. 346–351, Morgan Kaufmann Publishers Inc., 1995.

[103] J. MacCormick and A. Blake, "A probabilistic exclusion principle for tracking multiple objects," *International Journal of Computer Vision*, vol. 39, no. 1, pp. 57–71, 2000.

[104] I. M. Rekleitis, "A particle filter tutorial for mobile robot localization," *Centre for Intelligent Machines, McGill University, Tech. Rep. TR-CIM-04-02*, 2004.

[105] A. Turner and A. Penn, "Encoding natural movement as an agent-based system: an investigation into human pedestrian behaviour in the built environment," *Environ Plann B*, vol. 29, no. 4, pp. 473–490, 2002.

[106] M. Bierlaire, G. Antonini, and M. Weber, "Behavioral dynamics for pedestrians," in *Moving Through Nets: The Physical and Social Dimensions of Travel. 10th International Conference on Travel Behavior Research. Lucerne*, 2003.

[107] M. Batty, J. Desyllas, and E. Duxbury, "The discrete dynamics of small-scale spatial events: agent-based models of mobility in carnivals and street parades," *International Journal of Geographical Information Science*, vol. 17, no. 7, pp. 673–697, 2003.

[108] S. Gwynne, E. Galea, M. Owen, P. Lawrence, and L. Filippidis, "A review of the methodologies used in the computer simulation of evacuation from the built environment," *Building and Environment*, vol. 34, no. 6, pp. 741–749, 1999.

[109] N. A. Heck and D. B. Webster, "Wilderness Area Simulation Model - User's Manual. - Final rept.," Technical report PB-233 364/9, Forest Service, Washington, D.C., Jun 1973.

[110] J. VAN WAGTENDONK, "The wilderness simulation model," *International Journal of Wilderness*, vol. 9, no. 2, p. 9, 2003.

[111] G. Antonini, M. Bierlaire, and M. Weber, "Discrete choice models of pedestrian walking behavior," *Transportation Research Part B: Methodological*, vol. 40, no. 8, pp. 667–687, 2006.

[112] F. Johansson, "Microscopic modeling and simulation of pedestrian traffic," 2013.

[113] N. Pelechano and A. Malkawi, "Evacuation simulation models: Challenges in modeling high rise building evacuation with cellular automata approaches," *Automation in construction*, vol. 17, no. 4, pp. 377–385, 2008.

[114] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 261–268, IEEE, 2009.

[115] D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features of escape panic," *Nature*, vol. 407, no. 6803, pp. 487–490, 2000.

[116] D. Helbing, I. J. Farkas, P. Molnar, and T. Vicsek, "Simulation of pedestrian crowds in normal and evacuation situations," *Pedestrian and evacuation dynamics*, vol. 21, no. 2, pp. 21–58, 2002.

[117] N. Pelechano, J. M. Allbeck, and N. I. Badler, "Controlling individual agents in high-density crowd simulation," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 99–108, Eurographics Association, 2007.

[118] H. Xi, S. Lee, and Y.-J. Son, "An integrated pedestrian behavior model based on extended decision field theory and social force model," in *Human-in-the-Loop Simulations*, pp. 69–95, Springer, 2011.

[119] D. Helbing, P. Molnar, I. J. Farkas, and K. Bolay, "Self-organizing pedestrian movement," *Environment and planning B*, vol. 28, no. 3, pp. 361–384, 2001.

[120] M. Gardner, "Mathematical games: The fantastic combinations of john conways new solitaire game life," *Scientific American*, vol. 223, no. 4, pp. 120–123, 1970.

[121] S. Wolfram, "Statistical mechanics of cellular automata," *Reviews of modern physics*, vol. 55, no. 3, p. 601, 1983.

[122] A. Schadschneider, "Cellular automaton approach to pedestrian dynamics-theory," *arXiv preprint cond-mat/0112117*, 2001.

[123] C. Burstedde, A. Kirchner, K. Klauck, A. Schadschneider, and J. Zittartz, "Cellular automaton approach to pedestrian dynamics-applications," *arXiv preprint cond-mat/0112119*, 2001.

[124] C. Gloor, P. Stucki, and K. Nagel, "Hybrid techniques for pedestrian simulations," *Cellular Automata*, pp. 581–590, 2004.

[125] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," vol. 21, pp. 25–34, ACM, 1987.

[126] R. Gimblett, T. Daniel, S. Cherry, and M. Meitner, "The simulation and visualization of complex human–environment interactions," *Landscape and Urban Planning*, vol. 54, no. 1, pp. 63–79, 2001.

[127] C. Loscos, D. Marchal, and A. Meyer, "Intuitive crowd behavior in dense urban environments using local laws," in *Theory and Practice of Computer Graphics, 2003. Proceedings*, pp. 122–129, IEEE, 2003.

[128] J. J. Gibson, *The ecological approach to visual perception*. Routledge, 1986.

[129] F. Feurtey, "Simulating the collision avoidance behavior of pedestrians," *Master's Thesis*, 2000.

[130] K. Kitazawa and T. Fujiyama, "Pedestrian vision and collision avoidance behavior: Investigation of the information process space of pedestrians using an eye tracker," in *Pedestrian and evacuation dynamics 2008*, pp. 95–108, Springer, 2010.

[131] R. Pakbaz, A. A. Gohari, and V. Rodoplu, "TraJECT-3D: Generating realistic mobility traces for tactical network simulation," in *MILITARY COMMUNICATIONS CONFERENCE, 2011-MILCOM 2011*, pp. 967–972, IEEE, 2011.

[132] D. O'Sullivan and A. Turner, "Visibility graphs and landscape visibility analysis," *International Journal of Geographical Information Science*, vol. 15, no. 3, pp. 221–237, 2001.

[133] A. Turner, M. Doxa, D. O'sullivan, and A. Penn, "From isovists to visibility graphs: a methodology for the analysis of architectural space," *Environ Plann B*, vol. 28, no. 1, pp. 103–121, 2001.

[134] A. E. Minetti, C. Moia, G. S. Roi, D. Susta, and G. Ferretti, "Energy cost of walking and running at extreme uphill and downhill slopes," *Journal of Applied Physiology*, vol. 93, no. 3, pp. 1039–1046, 2002.

[135] W. Tobler, "Non-isotropic geographic modeling," *Three presentations on geographic analysis and modeling. Santa Barbara: National Center for Geographic Information and Analysis, University of California*, 1993.

[136] "Wikipedia." http://en.wikipedia.org/wiki/KullbackLast accessed 21 February, 2013.

[137] M. J. Swain and D. H. Ballard, "Color indexing," *International journal of computer vision*, vol. 7, no. 1, pp. 11–32, 1991.

[138] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International journal of computer vision*, vol. 40, no. 2, pp. 99–121, 2000.

[139] E. Levina and P. Bickel, "The earth mover's distance is the Mallows distance: some insights from statistics," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 2, pp. 251–256, IEEE, 2001.

[140] Y. Rubner, C. Tomasi, and L. J. Guibas, "A metric for distributions with applications to image databases," in *Computer Vision, 1998. Sixth International Conference on*, pp. 59–66, IEEE, 1998.

[141] F. L. Hitchcock, "The distribution of a product from several sources to numerous localities," *J. Math. Phys*, vol. 20, no. 2, pp. 224–230, 1941.

[142] S. Luke, "Multiagent simulation and the mason library," *George Mason University,*, 2011.

[143] E. J. Rykiel, "Testing ecological models: the meaning of validation," *Ecological modelling*, vol. 90, no. 3, pp. 229–244, 1996.

[144] F. Klügl, "A validation methodology for agent-based simulations," in *Proceedings of the 2008 ACM symposium on Applied computing*, pp. 39–43, ACM, 2008.

[145] R. T. Johnson, T. A. Lampe, and S. Seichter, "Calibration of an agent-based simulation model depicting a refugee camp scenario," in *Simulation Conference (WSC), Proceedings of the 2009 Winter*, pp. 1778–1786, IEEE, 2009.

[146] J. L. Berrou, J. Beecham, P. Quaglia, M. A. Kagarlis, and A. Gerodimos, "Calibration and validation of the Legion simulation model using empirical data," in *Pedestrian and Evacuation Dynamics 2005*, pp. 167–181, Springer, 2007.

[147] S. A. Brueckner and H. Van Dyke Parunak, "Resource-aware exploration of the emergent dynamics of simulated systems," in *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pp. 781–788, ACM, 2003.

[148] B. Calvez and G. Hutzler, "Ant colony systems and the calibration of multi-agent simulations: a new approach," in *Multi-Agents for modelling Complex Systems (MA4CS'07) Satellite Workshop of the European Conference on Complex Systems 2007 (ECCS'07)*, p. 16, 2007.

[149] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan, "An introduction to mcmc for machine learning," *Machine learning*, vol. 50, no. 1-2, pp. 5–43, 2003.

[150] C. Andrieu, A. Doucet, and R. Holenstein, "Particle markov chain monte carlo methods," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 72, no. 3, pp. 269–342, 2010.

[151] "Experiment." https://en.wikipedia.org/wiki/Experiment, 7 March. 2016.

[152] "Nist/sematech e-handbook of statistical methods," 10 March. 2013.

[153] J. P. Kleijnen, S. M. Sanchez, T. W. Lucas, and T. M. Cioppa, "State-of-the-art review: a users guide to the brave new world of designing simulation experiments," *INFORMS Journal on Computing*, vol. 17, no. 3, pp. 263–289, 2005.

[154] S. M. Sanchez and T. W. Lucas, "Exploring the world of agent-based simulations: simple models, complex analyses: exploring the world of agent-based simulations: simple models, complex analyses," in *Proceedings of the 34th conference on Winter simulation: exploring new frontiers*, pp. 116–126, Winter Simulation Conference, 2002.

[155] J. Kleijnen, S. Sanchez, T. Lucas, and T. Cioppa, "A userss guide to the brave new world of designing simulation experiments. tilburg university," *Center for Economic Research Discussion Paper*, no. 1, 2003.

[156] A. V. Noordegraaf, M. Nielen, and J. P. Kleijnen, "Sensitivity analysis by experimental design and metamodelling: Case study on simulation in national animal disease control," *European Journal of Operational Research*, vol. 146, no. 3, pp. 433–443, 2003.

[157] J. P. Kleijnen, "An overview of the design and analysis of simulation experiments for sensitivity analysis," *European Journal of Operational Research*, vol. 164, no. 2, pp. 287–300, 2005.

[158] K. Happe, "Agent-based modelling and sensitivity analysis by experimental design and metamodelling: an application to modelling regional structural change," in *XIth International Congress of the European Association of Agricultural Economists*, 2005.

[159] R. A. Lordo, "Simulation: A Statistical Perspective," *Technometrics*, vol. 35, no. 4, pp. 453–454, 1993.

[160] O. Cappé, S. J. Godsill, and E. Moulines, "An overview of existing methods and recent advances in sequential monte carlo," *Proceedings of the IEEE*, vol. 95, no. 5, pp. 899–924, 2007.

[161] J. Olsson, O. Cappé, R. Douc, E. Moulines, *et al.*, "Sequential monte carlo smoothing with application to parameter estimation in nonlinear state space models," *Bernoulli*, vol. 14, no. 1, pp. 155–179, 2008.

[162] N. Kantas, A. Doucet, S. S. Singh, and J. M. Maciejowski, "An overview of sequential monte carlo methods for parameter estimation in general state-space models," in *15th*

IFAC Symposium on System Identification (SYSID), Saint-Malo, France.(invited paper), vol. 102, p. 117, 2009.

[163] C. Andrieu, A. Doucet, and R. Holenstein, "Particle markov chain monte carlo methods," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 72, no. 3, pp. 269–342, 2010.

[164] H. F. Lopes and R. S. Tsay, "Particle filters and bayesian inference in financial econometrics," *Journal of Forecasting*, vol. 30, no. 1, pp. 168–209, 2011.

[165] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian data analysis*. Chapman Hall/CRC, London, 2nd edn ed., 2004.

[166] R. Casarin, "Monte carlo methods using matlab," 2011.

[167] N. Chopin, P. E. Jacob, and O. Papaspiliopoulos, "Smc2: an efficient algorithm for sequential analysis of state space models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 75, no. 3, pp. 397–426, 2013.

[168] N. Chopin, "A sequential particle filter method for static models," *Biometrika*, vol. 89, no. 3, pp. 539–552, 2002.

[169] N. J. Gordon, D. J. Salmond, and A. F. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," in *IEE Proceedings F (Radar and Signal Processing)*, vol. 140, pp. 107–113, IET, 1993.

[170] J. S. Rosenthal *et al.*, "Optimal proposal distributions and adaptive mcmc," *Handbook of Markov Chain Monte Carlo*, pp. 93–112, 2011.

[171] D. C. Parker, S. M. Manson, M. A. Janssen, M. J. Hoffmann, and P. Deadman, "Multi-agent systems for the simulation of land-use and land-cover change: a review," *Annals of the association of American Geographers*, vol. 93, no. 2, pp. 314–337, 2003.

[172] O. Balci, "Verification validation and accreditation of simulation models," in *Proceedings of the 29th conference on Winter simulation*, pp. 135–141, IEEE Computer Society, 1997.

[173] R. G. Sargent, "Verification, validation, and accreditation: verification, validation, and accreditation of simulation models," in *Proceedings of the 32nd conference on Winter simulation*, pp. 50–59, Society for Computer Simulation International, 2000.

[174] R. G. Sargent, "Verification and validation of simulation models," in *Proceedings of the 37th conference on Winter simulation*, pp. 130–143, winter simulation conference, 2005.

[175] "SAR Spotlight Forum." http://www.intermap.com/elevation-data. Last accessed 25th September, 2010.

[176] J. D. Hol, "Resampling in particle filters," 2004.

[177] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertin spatial relationships in robotics," *In Robotics and Automation Magazine IEEE. Proc. 1987 IEEE International Conference on*, vol. 4, pp. 850–851, 1987.

[178] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *The international journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986.

[179] H. F. Durrant-Whyte, "Uncertain geometry in robotics," *Robotics and Automation, IEEE Journal of*, vol. 4, no. 1, pp. 23–31, 1988.

[180] C. Harris and M.Stephens, "A Combined Corner and Edge Detection," *Proceedings of The Fourth Alvey Vision Conference*, pp. 147–151, 1988.

[181] P.Moreels and P.Perona, "Evaluation of features detectors and descriptors based on 3D objects," *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 1, pp. 800–807 Vol. 1, Oct. 2005.

[182] V.Lepetit and P.Fua, "Keypoint recognition using randomized trees," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 9, pp. 1465–1479, 2006.

[183] M.Bravo and H.Farid, "Object recognition in dense clutter," *Visual Cognition*, vol. 10, pp. 471–491, 2006.

[184] M. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *Robotics and Automation, IEEE Transactions on*, vol. 17, no. 3, pp. 229–241, 2001.

[185] A. J. Davison and N. Kita, "3d simultaneous localisation and map-building using active vision for a robot moving on undulating terrain," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, pp. I–384, IEEE, 2001.

[186] M. Bryson and S. Sukkarieh, "Bearing-only SLAM for an airborne vehicle," *In Australasian conference on robotics and automation, ACRA, Sydney*, 2005.

[187] M. T. Bryson and S. Sukkarieh, "Building a Robust Implementation of Bearing-Only Inertial SLAM for a UAV," *Field Robotics*, vol. 24, pp. 113–143, February 2007.

[188] W. Mohibullah and S. Julier, "Bearing-Only Localisation of Targets from Low-Speed UAVs," *FUSION*, 2010.

[189] J. Kim and S.Sukkarieh, "Autonomous airborne navigation in unknown terrain environments," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 40, no. 3, pp. 1031–1045, 2004.

[190] D. Ribas, P. Ridao, J. Neira, and J. D. Tardos, "Slam using an imaging sonar for partially structured underwater environments," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 5040–5045, IEEE, 2006.

[191] D. Ribas, P. Ridao, J. D. Tardós, and J. Neira, "Underwater slam in a marina environment," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 1455–1460, IEEE, 2007.

[192] N. Fairfield, G. Kantor, and D. Wettergreen, "Real-time slam with octree evidence grids for exploration in underwater tunnels," *Journal of Field Robotics*, vol. 24, no. 1, pp. 3–22, 2007.

[193] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, (Acapulco, Mexico), IJCAI, 2003.

[194] O.Ozisik and S. Yavuz, "An occupancy grid based SLAM method," *Computational Intelligence for Measurement Systems and Applications, 2008. CIMSA 2008. 2008 IEEE International Conference on*, pp. 117 – 119, 2008.

[195] S. Thrun and M. Montemerlo, "The graph SLAM algorithm with applications to large-scale mapping of urban structures," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–429, 2006.

[196] T.Bailey, "Constrained initialisation for bearing-only SLAM," *in Procceedings of IEEE Int. Conf. Robotics Automation*, vol. 2, pp. 1966–1971, Sept. 2003.

[197] J. Guivant, E. Nebot, J. Nieto, and F. Masson, "Navigation and mapping in large unstructured environments," *The International Journal of Robotics Research*, vol. 23, no. 4-5, pp. 449–472, 2004.

[198] J.Nieto, "DenseSLAM: Simultaneous localization and dense mapping," *International Journal of Robotics Research*, vol. 25, pp. 711–744, 2006.

[199] K. Pathak, A. Birk, N. Vaskevicius, M. Pfingsthorn, S. Schwertfeger, and J. Poppinga, "Online three-dimensional SLAM by registration of large planar surface segments and closed-form pose-graph relaxation," *Journal of Field Robotics*, vol. 27, no. 1, pp. 52–84, 2010.

[200] A. Kim and R. Eustice, "Pose-graph visual SLAM with geometric model selection for autonomous underwater ship hull inspection," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 1559–1565, IEEE, 2009.

[201] P. C. Hew, "The Linear Kalman Filter," 1997.

[202] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping part I," *IEEE Robotics & Automation Magazine*, vol. 13, pp. 99 – 110, 2006.

[203] J. Artieda, J. M. Sebastian, P. Campoy, J. F. Correa, I. F. Mondragón, C. Martínez, and M. Olivares, "Visual 3-D SLAM from UAVs," *Journal of Intelligent and Robotic Systems*, vol. 55, pp. 299–321, August 2009.

[204] S.J.Julier and J.K.Uhlmann, "A counter example to the teory of simultaneous localisation map building.," *In Robotics and Automation, 2001. Proc. 2001 ICRA. IEEE International Conference on*, vol. 4, pp. 4238–4243, 2001.

[205] A.J.Davison, "Real-time simultaneous localisation and mapping with a single camera," *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pp. 1403–1410 vol.2, Oct. 2003.

[206] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 6, pp. 1052–1067, 2007.

[207] R. Steffen and W. Förstner, "On Visual Real Time Mapping for Unmanned Aerial Vehicles," *21st Conference of the International Society for Photogrammetry and Remote Sensing (ISPRS)*, vol. XXXVII, no. Part B3a, pp. 57–62, 2008.

[208] J.Sola, A. Monin, M.Devy, and T.Lemaire, "Undelayed initialization in bearing only SLAM," *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 2499–2504, Aug. 2005.

[209] N. Kwok and G.Dissanayake, "An efficient multiple hypothesis filter for bearing-only SLAM," *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 1, pp. 736–741 vol.1, Sept.-2 Oct. 2004.

[210] N. Kwok, G.Dissanayake, and Q. Ha, "Bearing-only SLAM Using a SPRT Based Gaussian Sum Filter," *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 1109–1114, April 2005.

[211] J. Montiel, J. Civera, and A.Davison, "Unified Inverse Depth Parametrization for Monocular SLAM," *in Proceedings of Robotics: Science and Systems*, pp. 16–19, August 2006. This is a prestigious new single-track international robotics conference with online-only proceedings.

[212] J.Civera, A. Davison, and J. M. M. Montiel, "Inverse Depth Parametrization for Monocular SLAM," *IEEE Transactions on Robotics*, vol. 24, pp. 932–945, October 2008.

[213] P. Pinies, T. Lupton, S.Sukkarieh, and J. Tardos, "Inertial Aiding of Inverse Depth SLAM using a Monocular Camera," *Robotics and Automation, 2007 IEEE International Conference*, pp. 2797–2802, April 2007.

[214] M. Parsley and S. Julier, "Avoiding Negative Depth in Inverse Depth Bearing-Only SLAM," *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2066 – 2071, 2008.

[215] R. Munguia and A. Grau, "Delayed Features Initialization for Inverse Depth Monocular SLAM," *In European Conference on Mobile Robots*, pp. 1–6, 2007.

[216] J. Solà, "Consistency of the Monocular EKF-SLAM Algorithm for Three Different Landmark Parametrizations," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA2010)*, pp. 3–8, 2010.

[217] J. Nieto, J. E. Guivant, E. M. Nebot, *et al.*, "The hybrid metric maps (hymms): A novel map representation for denseslam," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 1, pp. 391–396, IEEE, 2004.

[218] J. Guivant, J. Nieto, F. Masson, and E. Nebot, "Navigation and mapping in large unstructured environments," *The International Journal of Robotics Research*, vol. 23, pp. 4–5, 2004.

[219] K. Sabe, M. Fukuchi, J.-S. Gutmann, T. Ohashi, K. Kawamoto, and T. Yoshigahara, "Obstacle avoidance and path planning for humanoid robots using stereo vision," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 1, pp. 592–597, IEEE, 2004.

[220] V. J. Blue and J. L. Adler, "Emergent fundamental pedestrian flows from cellular automata microsimulation," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1644, no. 1, pp. 29–36, 1998.

[221] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Fluids Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[222] D. Salmond, "Target tracking: introduction and kalman tracking filters," in *Target Tracking: Algorithms and Applications (Ref. No. 2001/174), IEE*, pp. 1–1, IET, 2001.

[223] F. Daum, "Nonlinear filters: beyond the kalman filter," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 20, no. 8, pp. 57–69, 2005.

[224] A. H. Jazwinski, *Stochastic processes and filtering theory*. Courier Corporation, 2007.

[225] S. J. Julier and J. K. Uhlmann, "A general method for approximating nonlinear transformations of probability distributions," tech. rep., Technical report, RRG, Department of engineering science, University of Oxford, 1996.

[226] S. J. Julier and J. K. Uhlmann, "New extension of the kalman filter to nonlinear systems," in *AeroSense'97*, pp. 182–193, International Society for Optics and Photonics, 1997.

[227] J. Kim, *Autonomous Navigation Airborn Applications*. PhD thesis, Austrilian Centre for Field Robotics, 2004.

[228] J. Kim and S.Sukkarieh, "Airborne simultaneous localisation and map building," *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference*, vol. 1, pp. 406–411 vol.1, Sept. 2003.

[229] E. Imre, M.-O. Berger, and N. Noury, "Improved Inverse-Depth Parameterization for Monocular Simultaneous Localization and Mapping," *IEEE International Conference on Robotics and Automation*, pp. 381–386, 2009.

[230] N. Sünderhauf, S. Lange, and P. Protzel, "Using the Unscented Kalman Filter in Mono-SLAM with Inverse Depth Parametrization for Autonomous Airship Control," *Proceedings of the IEEE International Workshop on Safety, Security and Rescue Robotics (SSRR 2007)*, pp. 1–6, September 2007.

[231] J. M. Passerieux and D. van Cappel, "Optimal Observer Maneuver for Bearings-Only Tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, pp. 777–788, July 1998.

[232] D. Schuhmacher, B. T. Vo, and B. N. Vo, "A Consistent Metric for Performance Evaluation of Multi-Object Filters," *IEEE Transactions on Signal Processing*, vol. 56, pp. 3447–3457, August 2008.

[233] P. T. Liu, "An optimum approach in target tracking with bearing measurements," *Journal of Optimization Theory and Applications*, vol. 56, pp. 205–214, February 1988.

[234] J. A. Fawcett, "Effects of Course Maneuvers on Bearing Only Range Estimation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, pp. 1193–1199, August 1998.